PROYECTO FIN DE CARRERA

Universidad Carlos III de Madrid

Ingeniería Técnica de Telecomunicación:

Telemática

Título: Creación de Cuestionarios en Editor XML/Gráfico basado en Eclipse RCP.

Autor: Jesus Domínguez Vicioso

Tutor: Mª Carmen Fernández Panadero.

EL TRIBUNAL

Presidente: Carlos Jesús Bernardos Cano

Secretario: Julio Villena Román

Vocal: Ángel García Olaya

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 23 de octubre de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo: Presidente Fdo: Secretario Fdo: Vocal

Agradecimientos

A mis padres, a quienes les debo todo y a los que ni en tres vidas podría agradecer todo lo que han dado y luchado para que yo llegue hasta aquí. Siempre seréis mis dos dioses y las personas más importantes de mi vida.

A Ali, porque sin ella esto no hubiese sido posible. Te estaré siempre agradecido por todo lo bueno que me has aportado. Te quiero mucho Winnie!

A mi hermana y mi cuñado, por estar ahí, ser vosotros y quererme tanto. Sabéis que el sentimiento es mutuo, os quiero.

A Luisi, Antonio padre e hijo, por ser como mis padres y hermano en Madrid.

A mi tía Mari, por estar siempre ahí, pendiente de todo. Te quiero tita!

A mi familia en general, abuelos que estarían orgullosos y a mi abuela que sé que lo está. A mis tíos, paternos y maternos, por todo su cariño desde que era un crio. A mis primos, pequeños y mayores, por regalarme tan buenos momentos.

A mis once valientes de Jaraíz, por aquel fin de semana inolvidable, único e irrepetible. Sois muy grandes!

A mis compañeros de la universidad, especialmente a Marta y Sergio por compartir conmigo tantísimas horas de prácticas. A David, por ser mi primer jefe y junto a Sergio, mis grandes amigos en los madriles. A Carol, Gloria y Ruth, por tantísimas horas de estudio y junto a Ali y Marta, por ayudarme tanto en la Uni.

A mis compañeros de trabajo, desde beca-Lega hasta GMV-Toulouse, pasando por beca-TID, Movidilo y GMV-Madrid. Por las horas de curro, las casas rurales, partidos de fútbol, cenas navideñas, cañas y buenos ratos.

A todos y cada uno de mis amigos. A mi gente de Valdastillas, Peri, Ale, Ernes y Vane, de Villanueva, Juanma, Carlitos, Javi, Rocío, Leti, Leo, Flo y Afteragüers en general, a los piornos, Alex, Miguel, Manuel, Carlos y un largo etcétera, a los campusos, Víctor, Carlos, Rubén, Ana, María…, a Alba, Isa y demás amig@s de otros lugares. Gracias por estar o haber estado en los momentos importantes.

A todos mis profesores y compañeros de clase, desde párvulos hasta ahora, desde Valdastillas hasta Leganés, pasando por Navaconcejo, Plasencia y Salamanca, por educarme y enseñarme a ser yo mismo.

Y de manera muy especial a Mari Carmen, mi tutora del proyecto fin de carrera. Sin su trabajo y dedicación jamás habría sido posible conseguir este objetivo. Han sido meses de duro trabajo pero ha merecido la pena. Espero no perder el contacto y le deseo de corazón toda la suerte del mundo. Mil gracias!

Resumen

El e-Learning permite la educación y capacitación a través de Internet. Cuando hablamos de e-Learning, hablamos de la posibilidad que ofrecen los sistemas informáticos de crear cuestionarios interactivos, representarlos sobre cualquier plataforma o tecnología, acceder a ellos independientemente de donde nos encontremos y responderlos en cualquier momento, además de la posibilidad de ver los resultados o crear estudios, estadísticas e informes acerca de ellos. Una de las principales características es que gracias a los estándares existentes, estos sistemas permiten intercambiarse información entre ellos.

Hoy en día, existen multitud de aplicaciones web que ofrecen al usuario opciones para crear sus propios cuestionarios, compartirlos, contestarlos, puntuarlos, visualizar los resultados, etc. Suelen estar integrados en sistemas LMS (Moodle o Dokeos) o sistemas para la edición de MOOCs (EDX), y la gran mayoría utiliza XML como lenguaje de representación de todas esa información. Sin embargo, se trata de sistemas muy acoplados y dependientes de su propia tecnología.

Este proyecto separa la definición y edición de cuestionarios de lo que sería la parte relativa a la presentación e interacción con el usuario final. Consiste en una aplicación que incluye un editor de ficheros XML (que van a ser los cuestionarios en sí), acompaño de un editor gráfico que permite modificar y pre-visualizar de manera sencilla y rápida cómo queda el cuestionario.

El entorno se ha desarrollado bajo la tecnología Eclipse RCP, que permite definir una aplicación de escritorio desde cero a la cual se la va añadiendo funcionalidad. De esta forma, se consigue un entorno que no va a depender de acceso a la red, salvo a la hora de compartir los proyectos de test generados.

Abstract

E-Learning is learning using electronic technologies to access educational outside of a traditional classroom. If we talk about "e-learning", we talk about the technology that allows creating interactive questionnaires, displaying them in different applications, access them remotely, access them regardless of where we are and answer them at any time. Furthermore, it offers the possibility to display the results, or create statistics, create reports…One of the main features is that e-learning is standardized, so these systems allow share information between them.

Actually, there are a lot of web applications that offers multiple choices to create their own questionnaires. Furthermore, these applications allow answer the questionnaires, put a score, show the results…These applications usually are integrated in LMS systems (Moodle or Blackboard) or systems to edit MOOCs (EDX). The vast majority of them use XML as language to represent the information. However, these systems are very dependent of their own technology.

This project separates the definition and edition from the presentation to the users. It consists of an application that included a XML editor (the questionnaires are XML files), a graphic editor to edit and display the questionnaires. It is an easy way to show the questionnaires to the users.

Eclipse RCP has been the technology used to develop the environment. Eclipse RCP allows defining an easy desktop application and adds new functionality module by module. Thus, it gets an independent environment. This environment doesn't depend on the network; it only needs the network when we share the test projects.

Índice General

Índice de Figuras

Índice de Tablas

Acrónimos

- **API**: Application Programming Interface
- **AWT**: Abstract Window Toolkit
- **CSS**: Cascading Style Sheets
- **CVS**: Concurrent Version System
- **DnD**: Drag and Drop
- **DOM**: Document Object Model
- **DTD**: Document Type Definition
- **GTK**: GIMP Toolkit
- **GUI**: Graphical User Interface
- **HTML**: HyperText Markup Language
- **IDE**: Integrated Development Environment
- **Java SE**: Java Standard Edition
- **JAXB**: Java Architecture for XML Binding
- **JDK**: Java Development Kit
- **JDT**: Java Development Tool
- **JNI**: Java Native Interface
- **JRE**: Java Runtime Environment
- **JVM**: Java Virtual Machine
- **LMS**: Learning Management System
- **MOOC**: Massive Open Online Course
- **MOODLE**: Modular Object-Oriented Dynamic Learning Environment
- **MVC**: Modelo Vista Controlador
- **QTI**: Question and Test Interoperability

- **RCP**: Rich Client Platform
- **SCORM**: Sharable Content Object Reference Model
- **SOAP**: Simple Object Access Protocol
- **SVN**: Subversion
- **SWT**: Standard Widget Toolkit
- **W3C**: World Wide Web Consortium
- **XML**: eXtensible Markup Language
- **XSD**: XML Schema Definition

Capítulo 1

Introducción

La definición de e-Learning

[1] corresponde con la educación y capacitación a través de Internet. Este tipo de enseñanza online permite la interacción del usuario con el material mediante la utilización de diversas herramientas informáticas.

Aunque la mayoría de estas herramientas e-Learning se suelen relacionar con el mundo docente, se debe destacar el importante papel de estas aplicaciones en el mundo empresarial, ya que facilitan al trabajador ampliar su formación de forma online, dándole una mayor flexibilidad horaria y ayudándole a mejorar la gestión de su tiempo. Los productos que se crean basados en estos estándares son productos que no se quedarán obsoletos en un corto periodo de tiempo, y esto hace que invertir en este tipo de herramientas sea un valor rentable.

El término "e-Learning" es la simplificación de Electronic Learning, y comprende fundamentalmente los siguientes aspectos:

- El **pedagógico**, referido a la Tecnología Educativa como disciplina de las ciencias de la educación, vinculada a los medios tecnológicos, la psicología educativa y la didáctica.

- El **tecnológico**, referido a la Tecnología de la Información y la Comunicación, mediante la selección, diseño, personalización, implementación, alojamiento y mantenimiento de soluciones en dónde se integran tecnologías propietarias y de código abierto (Open Source).

A primera vista, los componentes **tecnológicos** son los más tangibles, y el ejemplo más significativo son las plataformas de e-Learning o LMS, sistemas que permiten la administración y control de los aspectos administrativos de la capacitación entre otras funciones.

Los aspectos **pedagógicos** son el alma del e-Learning. Estos aspectos son los que trabajan sobre los contenidos. Aunque menos tangibles que los tecnológicos, finalmente serán los más relevantes ya que determinarán la eficacia de los objetivos y aprendizaje fijados. [1]

Cabe destacar como beneficios de las herramientas de e-Learning: [1]

- **Reducción de costes**: permite reducir y hasta eliminar gastos de traslado, alojamiento, material didáctico, etc.

- **Rapidez y agilidad**: Las comunicaciones a través de sistemas en la red confiere rapidez y agilidad a las comunicaciones.

- **Acceso "just-in-time"**: los usuarios pueden acceder al contenido desde cualquier conexión a Internet, cuando les surge la necesidad.

- **Flexibilidad de la agenda**: no se requiere que un grupo de personas coincidan en tiempo y espacio, sino que ofrece la ventaja que una vez publicado un curso, los consumidores de este curso, pueden realizarlo en cualquier franja horaria, se pueden fijar sus propios ritmos de aprendizaje… En definitiva ayuda a la gestión de los recursos del usuario.

- **Formación personalizada:** Los cursos de e-Learning ofrecen la ventaja de poder ser personalizados, de forma que a los usuarios se les mostrará información sobre sus progresos, información personalizada de las actividades realizadas, etc.

- **Actualización automática de contenidos:** Los usuarios acceden a información siempre actualizada, ya que la información se puede modificar en cualquier momento.

La aparición de los MOOCs [2] ha dado más relevancia aún si cabe a la evaluación a través que proporcionan los sistemas e-Learning, ya que permiten la creación de cursos online dirigidos a un amplio número de participantes a través de internet, siguiendo el principio de educación abierta.

De esta forma, la principal ventaja o una de las más importantes de este tipo de plataformas, es que permiten, de manera rápida y automática, la autocorrección de cuestionarios. Esta característica es muy importante a la hora de realizar ciertas correcciones a un elevado número de personas o también si los resultados han de obtenerse en un tiempo reducido.

La finalidad de este primer capítulo es además de introducir el sistema de e-Learning a alto nivel, las motivaciones que han llevado a desarrollar esta herramienta, los objetivos que se han fijado para este proyecto y explicar cómo se ha estructurado esta memoria.

A lo largo de esta memoria se irá evolucionando desde la necesidad, los objetivos desde los que se parte para iniciar este proyecto, un análisis y comparativa con otras herramientas actuales hasta terminar con la propuesta final de una nueva aplicación para facilitar la definición, edición y gestión. Para el desarrollo de esa herramienta se ha realizado un análisis sobre las características, requisitos y funcionalidades necesarias para una herramienta

óptima que cumpla las necesidades, que cubra las ideas que se plantean en este proyecto.

1.1 Motivación

Ante la constante evolución hacia un mundo en el que todo esté informatizado: realizar la compra semanal por Internet, seguimiento de las notas de los hijos en el colegio de forma online, trámites administrativos como la gestión de multas, renovación de documentación… cada vez existe un menor número de actividades que necesitan de la presencia física. Este hecho ayuda a que se pueda gestionar el tiempo de forma más eficiente y hace que cualquier tipo de actividad tienda a su virtualización.

Con estas premisas enfocadas en el entorno de la educación, surge la inquietud de conocer aplicaciones basadas en los estándares de e-Learning y estudiar sus ventajas y carencias para construir una herramienta más robusta que proporcione alguna ventaja que de un valor añadido a esta herramienta frente a otras.

Tras realizar un primer estudio acerca de este tipo de aplicaciones, nos encontramos con que la mayoría de los sistemas LMS, como por ejemplo **Moodle** [5] o **Dokeos** [7], disponen se sus propias herramientas para la edición y despliegue de cuestionarios. Dado que nuestra aplicación tiene como fin precisamente eso, la edición de cuestionarios, se han buscado puntos que puedan ser mejorables con respecto a esos sistemas, aun perdiendo funcionalidad y opciones pero potenciando otras, o también añadiendo por qué no funciones nuevas e innovadoras.

Por ello, este proyecto se centra en la creación de una herramienta de creación de test con las siguientes características:

- Interfaz gráfica amigable al usuario, intuitiva y de fácil manejo.
- Editor de test gráfico y de código fuente, que permita visualizar o modificar el fichero de test en formato XML de manera directa.
- Posibilidad de trabajar con la herramienta sin necesidad de disponer de conexión a la red.
- Control de versiones de los proyectos de test generados.
- Conexión a repositorios de contenido compartidos.
- Sistema modular, fácil de modificar o añadir nuevas funcionalidades.

El objetivo por tanto es implementar una aplicación de escritorio que disponga de una serie de funcionalidades que cubran todas esas características.

De inicio se ha realizado otro estudio acerca de este tipo de herramientas de escritorio, y se han encontrado algunas como **eXe Learning** [9] o **Hot Potatoes** [10] que a priori ofrecen características muy similares a las que busca otorgar esta aplicación.

Sin embargo, se encuentran puntos en los cuales se puede mejorar e incluso algunos en los que se puede innovar, como por ejemplo la posibilidad de conectar la aplicación a un repositorio de contenidos, que permita disponer de un control de versiones o incluso añada la posibilidad del trabajo en grupo.

Además, la elección de Eclipse RCP como tecnología a partir de la cual se va a desarrollar la aplicación, abre un abanico de posibilidades a la hora de decidir cómo se va desarrollar la misma. Son infinitas las tecnologías que RCP permite integrar en un solo entorno, y muchas de ellas se ajustan exactamente a las necesidades que requiere este proyecto.

1.2 Objetivos

El objetivo inicial de este proyecto es la creación de una herramienta de e-Learning que permita la creación, visualización y gestión de cuestionarios, desarrollada en una plataforma portable y robusta, mediante una interfaz fácil de usar y amigable de cara al usuario.

Como se ha introducido anteriormente, existen ya numerosas aplicaciones que permiten crear, editar y compartir cuestionarios, por lo tanto los objetivos se deben centrar en aportar algo diferente y cumplir con las características enumeradas anteriormente.

Para conseguir dotar a la aplicación de todas estas características, se han definido una serie de requisitos, funcionales y no funcionales.

Los requisitos funcionales marcados son:

- Crear una interfaz sencilla, amigable y fácil de usar, que no precise de usuarios avanzados con conocimientos de e-Learning u otro tipo de tecnologías.

- Posibilidad de utilizar la aplicación para crear o editar los test sin necesidad de disponer de conexión a la red.

- Se busca que el resultado a la salida de esta aplicación tenga un formato sencillo y fácil de entender, y que pueda ser importado

fácilmente en otras aplicaciones de e-Learning a desarrollar en un futuro.

- Posibilidad de configuración de repositorios compartidos, que den la posibilidad de trabajar en paralelo y desde diferentes sistemas.

- Disponer de un control de versiones, que permitan ver la evolución o volver a versiones anteriores de los proyectos de test.

- Posibilidad de generar diferentes versiones de la aplicación, en función del sistema operativo y/o el idioma configurado.

En cuanto a los requisitos no funcionales:

- Ha de ser un sistema modular, en el que cada una de las partes que lo componen pueda ser reusada, y en el cual sea sencillo introducir nuevas funcionalidades, así como mejorar las ya existentes. Se trata por tanto de separar los diferentes módulos que componen la aplicación, de manera que cada una de esas partes tenga por separado una función determinada, y todas en común lleguen a dar la funcionalidad esperada al entorno. Así, será posible sustituir cualquiera de ellas sin que las demás se vean afectadas, y también exportar la funcionalidad de cualquiera de ellas e incluirlas en otro sistema.

- El proyecto debe centrarse en la parte de la edición de los cuestionarios de los sistemas e-Learning, pero sin perder de vista que la solución propuesta tenga sentido en un sistema o entorno final. Se busca simplificar el diseño de cuestionarios, de manera que no resulten tan complicados de comprender por usuarios inexpertos.

- Otro de los requisitos de esta aplicación y uno en los que más hincapié se hará a la hora de desarrollar el código fuente, es que éste ha de ser estructurado, legible, bien documentado, que evite la duplicidad de código y fomente la reutilización, etc. En definitiva, que cumpla con las reglas de calidad mínimas definidas por los estándares.

1.3 Contenido de la memoria

Esta memoria consta de 8 capítulos y 5 anexos. En ella se explica el desarrollo, diseño, implementación y funcionalidad del proyecto fin de carrera desarrollado.

La memoria está organizada en los siguientes apartados principales:

En el capítulo 1 se detalla una "**Introducción**" acerca de los motivos que han generado el desarrollo de este proyecto y los objetivos que se persiguen cumplir con el mismo, y define qué es el e-Learning.

El segundo capítulo trata el "**Estado del arte**". En este apartado se habla de herramientas similares a la solución propuesta y se explican las tecnologías empleadas.

El capítulo 3 explica "**El diseño e implementación del sistema**". Define los módulos que componen el desarrollo, explica cómo se han estructurado e implementado éstos, y resume las pruebas efectuadas en el sistema.

En el capítulo 4 se detalla la "**Descripción de la aplicación**". En él se resume el funcionamiento de la aplicación, mediante diagramas y capturas de pantalla de la misma.

En el quinto capítulo se habla de la "**Gestión y planificación del proyecto**". Incluye la historia del proyecto, su ciclo de vida, organización, y por último se incluye el presupuesto de todo el desarrollo.

El capítulo número 6 trata los "**Resultados y conclusiones**", detallando si se han cubierto o no los requisitos marcados y los resultados obtenidos.

En el séptimo capítulo se enumeran una serie de "**Trabajos futuros**".

El último se trata de la **"Bibliografía"** usada.

En cuanto a los "**Anexos**", destacar que se incluye el manual de usuario de la aplicación, en el cual se detallan todas las opciones proporcionadas por la aplicación, el plan de pruebas, algunos ficheros de configuración y la explicación de los procedimientos para configurar Eclipse y crear un nuevo plugin.

Capítulo 2

Estado del arte

2.1 Introducción

En este apartado se analizan algunas herramientas existentes similares a la solución propuesta. Se citan tanto herramientas de creación de cuestionarios integradas en un LMS como herramientas externas. Dentro de los LMS se analizarán Moodle [5] y Dokeos [7], con sus principales características, haciendo especial hincapié en la parte de definición de cuestionarios, con sus ventajas e inconvenientes respecto a la solución planteada aquí. Dentro de las herramientas externas se analizarán exeLearning [9] y HotPotatos [10], que son dos de las más extendidas, y se compararán con la solución propuesta.

Además, en un segundo apartado se citarán todas las tecnologías utilizadas por esta aplicación, desde el lenguaje Java, pasando por XML, y continuando con el entorno de desarrollo Eclipse y todas las tecnologías y herramientas derivadas de este que hayan sido utilizadas en este proyecto. Se resumirán sus características principales y se hablará del uso que se les ha dado en el desarrollo de la aplicación.

2.2 Análisis de herramientas existentes

Este proyecto no se trata de un sistema novedoso, puesto que son muchos y muy variados las aplicaciones que permiten crear cuestionarios de forma interactiva y compartir estos con otros usuarios.

A continuación se citan algunos de ellos, se habla de las cosas positivas que aportan y las desventajas que tienen con respecto al sistema implementado en este proyecto.

Todas ellas soportan el formato **SCORM** [3], que se define como un conjunto de estándares y especificaciones que permite crear objetos pedagógicos estructurados, que pueden importarse dentro de sistemas de gestión de aprendizaje diferentes, siempre que estos soporten la norma SCORM. Sus principales características son la accesibilidad, adaptabilidad, durabilidad, interoperabilidad y reusabilidad.

Además de este formato, algunas van a soportar también el estándar **QTI** [4]. Al igual que SCORM, se usa para la representación de contenidos y resultados de evaluaciones educativas, soportando el intercambio de este material entre sistemas de creación y de visualización, repositorios y otros sistemas LMS. Permite la creación y entrega de materiales de pruebas en múltiples sistemas de forma intercambiable y fue diseñado para facilitar la interoperabilidad entre sistemas. La especificación consiste en un modelo de datos que define la estructura de preguntas, evaluaciones y resultados con una representación vinculada en XML que define esencialmente un lenguaje para intercambiar preguntas y otros materiales de evaluación.

Sin embargo ninguna de ellas contempla ni promueve el trabajo en grupo, colaborativo, ni dispone de un control de versiones que permita recuperar las diferentes versiones de las actividades realizadas. Esto es algo que sí cumplirá este sistema, ya que va a permitir disponer de repositorios de contenido remotos que proporcionen este tipo de herramienta.

Todas ellas argumentan la facilidad de uso como uno de sus puntos fuertes, pero se podría decir que eso no es del todo cierto, al menos no en la medida en la que este sistema quiere proporcionarle al usuario.

2.2.1 Herramientas Web

Este apartado analiza dos ejemplos de herramientas web que proporcionan servicio de e-Learning: **Moodle** y **Dokeos**. A estas tecnologías, se podrían añadir muchas otras que siguen esta misma filosofía, como pueden ser: Blackboard, Claroline, EDX o Chamilo, pero se ha decidido hablar sobre estas dos porque ambas se tratan de herramientas e-Learning muy conocidas y utilizadas.

Se hablará de sus principales características y el estudio se centrara sobre todo en la parte de edición de cuestionarios, mostrando las ventajas e inconvenientes que pueda haber con respecto a la solución implementada.

2.2.1.1 Moodle

Figura 1 - Icono de Moodle

(Fuente: https://moodle.org/)

Se trata una plataforma de aprendizaje diseñada para proporcionar a educadores, administradores y estudiantes un sistema integrado único, robusto y

seguro para crear ambientes de aprendizaje personalizados. Tiene un enfoque claramente pedagógico y es muy útil a la hora de llevar a cabo evaluaciones del curso. [5]

Está compuesto por multitud de herramientas, lo que le hace ser un sistema muy complejo. No obstante, es necesario resaltar que sólo pueden ser utilizadas conectados a Internet. Estas herramientas se dividen en módulos: tareas, consultas, foro, diario, recurso, encuesta, wiki, y la más importante en este caso, el módulo de cuestionarios.

Este módulo permite a los profesores definir una base de datos de preguntas que podrán ser reutilizadas en diferentes cuestionarios, las cuales pueden ser almacenadas en categorías de fácil acceso y estas categorías pueden ser "publicadas" para hacerlas accesibles desde cualquier curso del sitio. Los cuestionarios se califican automáticamente, y pueden ser recalificados si se modifican las preguntas, y los cuestionarios pueden tener un límite de tiempo a partir del cual no estarán disponibles. El profesor puede determinar si los cuestionarios pueden ser resueltos varias veces y si se mostrarán o no las respuestas correctas y los comentarios, las preguntas y las respuestas de los cuestionarios pueden ser mezclados (aleatoriamente) para disminuir las copias entre los alumnos. Las preguntas pueden crearse en HTML y con imágenes, pueden importarse desde archivos de texto externos, tener diferentes métricas y tipos de captura, etc. Además, soporta tanto SCORM como QTI, aunque esta última no por completo en su última versión. [6]

2.2.1.2 Dokeos

Figura 2 - Icono de Dokeos
(Fuente: http://www.dokeos.com/)

Es una aplicación de administración de contenidos de cursos, que reúne e integra todos los componentes necesarios para permitir la gestión, administración, comunicación, evaluación y seguimiento de las actividades de enseñanza y aprendizaje en la red. Entre sus principales características destaca que es de código abierto, modular, que permite agregar y modificar herramientas, adaptar bases de datos y sobre todo, que se trata de un sistema flexible y de muy fácil uso mediante una interfaz de usuario sumamente amigable. [7]

El siguiente mapa conceptual muestra las posibilidades que ofrece Dokeos, y en él se puede observar como la parte de test es solo una de los muchos útiles disponibles:

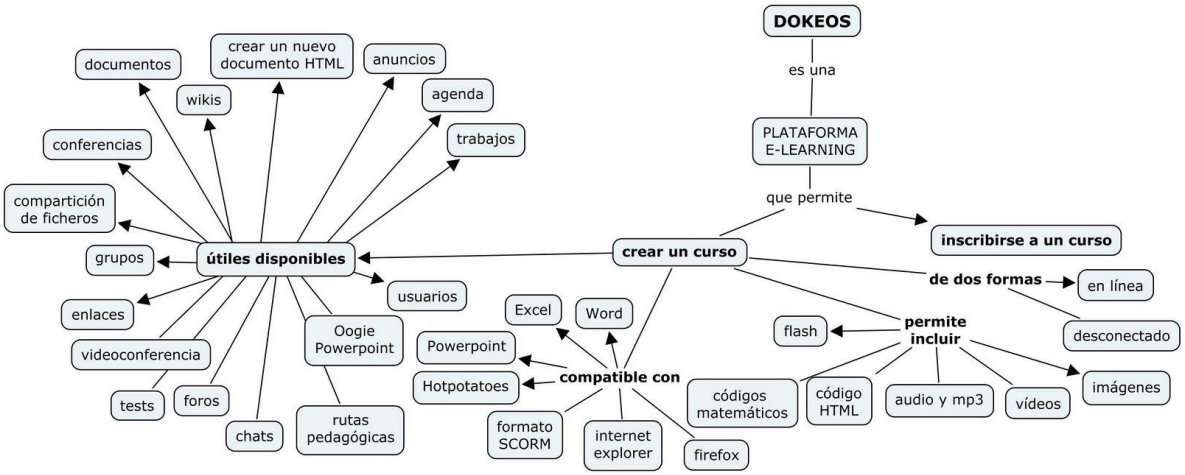


Figura 3 - Mapa conceptual de Dokeos
(Fuente: http://plataformadokeos.blogspot.fr/)

Esta parte, la de la utilidad de los test, es la que interesa en este caso. Una de sus principales características en el numeroso tipo de preguntas que pueden usarse, hasta un total de 29 diferentes, agrupadas en diferentes tipos de razonamiento lógico: preguntas con una sola respuesta válida o con múltiples respuestas correctas, respuestas libres, de inclusión de un fichero externo, de maridaje, de interacción con el punto del ratón, y un largo etcétera. [8]

En cuanto al cuestionario en general, se podrá definir la apariencia de este, de su encabezado o pie de página, incluyendo imágenes, logotipos, texto… También permite fijar un cronómetro que controle el tiempo del cual disponen los usuarios para contestar el test, configurar si el orden de las preguntas es siempre el mismo o puede ser aleatorio, si de todas esas preguntas de veden contestar todas o solo una selección, también evaluar las respuestas y poner comentarios, etc.

2.2.1.3 Conclusiones

La mayor dificultad que engloba este tipo de sistema es sin duda la complejidad que tienen, debido al gran número de opciones que ofrecen. Centrándose únicamente en la edición de cuestionarios, existen multitud de opciones, por ejemplo un número muy elevado de tipos de preguntas y respuesta, que para cualquier usuario, con o sin experiencia, puede resultar complicado de entender. Además, existe la restricción de que es necesario tener conexión a la red para poder usar estas aplicaciones.

A continuación se muestra en una tabla sobre cuáles son las ventajas e inconvenientes de las tecnologías descritas anteriormente, centrándose en la parte de creación de cuestionarios, con respecto a la solución propuesta por esta aplicación.

Tecnología	Ventajas	Inconvenientes
MOODLE	• Muy probado y de confianza • Gran número de tipo de preguntas y opciones • Siempre actualizado • Multitud de idiomas • Disponible en cualquier lugar con acceso a la red • Maquetación CSS • Gestión de formato SCORM y QTI: importación, edición y exportación	• Dependiente de la Red • Cuestionarios muy complejos • Complicado para usuarios sin experiencia • No dispone de control de versiones
DOKEOS	• Fácil creación y organización de contenidos interactivos y ejercicios • Gran número de tipo de preguntas y opciones • Facilidad de uso • Código disponible • Multitud de idiomas • Disponible en cualquier lugar con acceso a la red • Gestión de formato SCORM y QTI: importación, edición y exportación	• Dependiente de la Red • Cuestionarios complejos • No dispone de control de versiones

Tabla 1 – Ventajas e inconvenientes de Moodle y Dokeos respecto a la solución implementada

2.2.2 Herramientas de Escritorio

En este apartado, al contrario que el anterior, se habla solo y exclusivamente de aplicaciones de escritorio que permiten crear cuestionarios, y que por lo tanto comparten características con el sistema desarrollado por este proyecto. Concretamente se van a describir las principales características de **eXe Learning** y **Hot Potatoes**, dos de los entornos de creación de test más usados y similares a la solución desarrollada.

2.2.2.1 eXe Learning

Figura 4 – Icono de eXe Learning

(Fuente: http://exelearning.net/)

Se trata de un programa libre y abierto que ayuda a los docentes en la creación y publicación de contenidos, y que permite a profesores y académicos la publicación de contenidos didácticos en soportes informáticos (CD, memorias USB o en la web), sin necesidad de ser ni convertirse en expertos en HTML, XML o HTML5. Además, está disponible en GNU/Linux, Microsoft Windows y Mac OS X. [9]

Los recursos creados en eXe Learning son accesibles en formato XHTML o HTML5, pudiendo generarse sitios web completos (páginas web navegables), insertar contenidos interactivos (preguntas y actividades de diferentes tipos) en cada página, exportar los contenidos creados en otros formatos como ePub3 (libros electrónicos), QTI o SCORM, XLIFF (un estándar para la traducción) y catalogar los contenidos con diferentes modelos de metadatos (Dublin Core, LOM o LOM-ES). [9]

La aplicación se trata de un ejecutable que se abrirá en el navegador por defecto, aunque esto es configurable. De esta forma, su apariencia es muy similar a las de una página web, pero sin necesidad de estar conectado a la red. En ella existe un editor de ficheros de test, que permite pre-visualizar lo que se va creando. La maquetación se hace por columnas y no por tablas, y permite la inclusión de recursos gráficos como imágenes, audios o videos, en números formatos. Incluye JQuery y es posible exportar e importar estilos personalizados que incluyan JavaScript o código HTML personalizado.

Se trata por tanto de una tecnología muy avanzada, que permite multitud de opciones a la hora de crear y trabajar con ficheros de test.

2.2.2.2 Hot Potatoes

Figura 5 – Icono de Hot Potatoes

(Fuente: https://hotpot.uvic.ca/)

Se trata de una herramienta de escritorio compuesta por diferentes aplicaciones, denominadas patatas, que permite la creación de múltiples tipos de test y preguntas. [10]

En concreto, es posible crear hasta cinco tipos de test diferentes: [10]

- JQuizz: cuestionarios de elección múltiple, preguntas abiertas, híbridas o multi-selección.
- JCloze: rellenar huecos en frases o palabras.
- JMatch: Unir texto con texto, imágenes con texto, o texto con elementos multimedia.
- JMix: ordenar frases o letras de una palabra.
- JCross: crear crucigramas.

Los test generados podrán ser visualizados en la web, ya que se trata de ficheros con formato HTML, que no XML.

Las principales características de esta herramienta radican en: [11]

- simplicidad a la hora de crear o modificar los ficheros, ya que se trata de un sistema de fácil manejo e intuitivo.
- Aplicabilidad a la hora de usarse en cualquier entorno, educativo o no.
- Universalidad, en el sentido de que se pueden crear ficheros en diferentes idiomas fácilmente; también porque los ficheros se pueden compartir; y por último, por su gratuidad.

Se trata de un programa gratuito, aunque no de código abierto. Dispone de diferentes versiones: un ejecutable de Windows, otro de Linux, y una versión diferente denominada Java Hot Potatoes, que puede ser usada en cualquier sistema operativo, pero que no dispone de todas las opciones, como por ejemplo la de exportar a SCORM. [12]

En resumen, *Hot Potatoes* tiene múltiples aplicaciones, especialmente en el ámbito educativo, en el cual se puede emplear como material didáctico en cualquiera de las asignaturas. La novedad que supone su uso no es el tipo de ejercicios en sí, habituales en papel, sino el formato en que se presentan, HTML, el elevado número de preguntas posibles, y también la posibilidad de difusión por la red. [11]

2.2.2.3 Conclusiones

Ambos sistemas expuestos se tratan de herramientas que permiten crear y editar cuestionarios sin disponer de acceso a la red, archivos que cumplen con estándares reconocidos y que hacen que estas aplicaciones puedan llegar a ser muy útiles en el mundo e-Learning.

Sin embargo, eXe Learning se trata de un sistema complejo que permite multitud de opciones, que a su vez resultan muy complicadas de entender, sobre todo para usuarios inexpertos. Tampoco permite compartir los cuestionarios creados con un repositorio de contenidos, que además proporciones control de versiones.

En cuanto a Hot Potatoes, dispone de algún que otro punto débil, como puede ser la generación en HTML o que no permita compartir los test generados de manera automática. Sin embargo, se aproxima mucho a la solución buscada en este proyecto: interfaz sencilla y amigable, con opciones fáciles de encontrar.

A continuación se muestra una tabla donde se muestras las ventajas e inconvenientes de estas herramientas con respecto a la aplicación implementada.

Tecnología	Ventajas	Inconvenientes
eXe Learning	• Sistema muy probado • Sistema en continuo desarrollo • Código disponible • Aporta multitud de opciones • Soporte de SCORM y QTI • Maquetación de los test • Múltiples idiomas	• Sistema complejo • Gran número de opciones • Interfaz difícil de comprender • No dispone de control de versiones
Hot Potatoes	• Sistema muy probado • Soporte de SCORM y QTI • Aporta multitud de opciones • Múltiples idiomas • Facilidad de uso	• Código no disponible • Se trata de cinco aplicaciones por separado • No dispone de control de versiones

Tabla 2 – Ventajas e inconvenientes de eXe Learning y Hot Potatoes respecto a la solución implementada

2.3 Tecnologías empleadas

Por un lado, se habla de las características del metalenguaje **XML**, que es la tecnología usada para definir la estructura de los cuestionarios que se van a definir en la aplicación y que serán exportados a otros sistemas. Por otro lado, se incluye una pequeña introducción al lenguaje de programación utilizado a la hora de desarrollar la aplicación: **Java**.

Se hablará de las principales características de Eclipse, que no se trata únicamente del entorno utilizado para desarrollar la aplicación, sino que es su distribución **Eclipse RCP** sobre la cual se desarrolla este proyecto. También se explican las tecnologías aplicadas sobre Eclipse RCP, como son **SWT**, **JFace** y **Eclipse Forms**.

Además, se cita que se han añadido los plugins que dotan a la aplicación de la posibilidad de configurar repositorios compartidos, que a su vez soportan control de versiones de proyectos y ficheros. También se incluye un pequeño resumen de las utilidades empleadas para dar consistencia y calidad al código desarrollado, como son las opciones de '**formatter**', **'clean-up'** y '**checkstyle**' de programación. También se detalla cómo se ha procedido a la hora de **internacionalizar** la aplicación o dar la posibilidad de visualizar el entorno en diferentes idiomas.

2.3.1 XML

Se trata de un metalenguaje desarrollado por el W3C y que permite definir lenguajes de marcado. XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Es esencial en el mundo actual, en el que se tiende a la globalización y a la compatibilidad entre sistemas, ya que esta tecnología va a permitir compartir la información con seguridad y fiabilidad. [13]

Además, XML permite al programador dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corren a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello.

2.3.1.1 Objetivos

Los objetivos de diseño de XML son: [13]

- Directamente utilizable en Internet.
- Soportar una amplia variedad de aplicaciones.

- Compatible con SGML.
- Fácil procesamiento de lectura y escritura.
- Diseño formal y conciso.
- Extensible, para poder utilizarlo en el mayor ámbito posible.
- Trata de ser idéntico a HTML en cuanto al procesado de la información, para así aprovechar la tecnología implantada por este lenguaje.
- Fácil de implantar, programar y aplicar a los distintos sistemas.

2.3.1.2 Usos

En la actualidad XML es utilizado en: [14]

- Los sistemas de información permitiendo compartir y reutilizar contenidos, como por ejemplo en *e-Learning*.
- También es usado en *e-business* para poder representar transacciones comerciales.
- En computación distribuida entre sistemas (SOAP).
- Desarrollo de portales.
- Permite búsquedas más eficientes y orienta los contenidos a una lógica semántica.
- Facilita la accesibilidad a Internet a personas con discapacidad, en tanto que ofrece nuevos formatos como el "Voice XML".

2.3.1.3 Tecnologías XML

2.3.1.3.1 DTD

Describe la estructura que va a tener el documento XML al que se aplica esta DTD. Definen los elementos, atributos y entidades permitidas que va a tener un documento, define además el orden en el que tienen que ir los elementos y también nos va a indicar las limitaciones que tendrá cada elemento y la forma de anidarlos. Es como crear nuestro propio lenguaje de marcado para una aplicación específica. [15]

La DTD del XML es opcional ya que no todos los documentos XML requieren un DTD, pero los documentos que se ajustan a su DTD se denominan documentos "válidos".

2.3.1.3.2 XML Schema

Es un lenguaje de esquema utilizado para definir la estructura y limitaciones de los documentos XML que lo aplican. Está pensado para proporcionar una mayor potencia expresiva que las DTD. Al igual que las DTD, define un conjunto de reglas que debe cumplir el documento XML para ser considerado "válido", aumentando las posibilidades y funcionalidades de aplicaciones de procesado de datos. [14]

2.3.1.4 Uso dado en la aplicación

Se ha empleado XML para definir la estructura de los ficheros de test generados por la aplicación. El contenido de esos ficheros va ser un código estructurado compuesto de elementos predefinidos, de los cuales se hablará en apartados posteriores.

Estos ficheros serán generados y representados en un editor XML, que contará además con la ayuda de un analizador lógico basado en el *schema (fichero .xsd)* que define la estructura de este tipo de ficheros de test.

2.3.2 Java

Figura 6 - Icono de Java
(Fuente: https://www.java.net/)

Se trata de un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. [16]

2.3.2.1 Características

Las principales características de Java son: [16]

- **Es orientado a objetos**: control y organización de código.
- **Es muy flexible**: reutilización del código.

- **Multiplataforma:** es ejecutado por la JVM.
- **Código abierto**: librerías nativas a disposición de los desarrolladores.
- **Extensible**: puede ser extendido para mejorar el código nativo.
- **Gratuidad:** Su uso no acarrea inversiones económicas.
- **Recolector de basura**: evita en gran medida la pérdida de memoria.

2.3.2.2 Versiones

Un histórico de las diferentes versiones será: [17]

Nombre	Versión	Fecha de lanzamiento
JDK	1.0	23 de enero de 1996
JDK	1.1	19 de febrero de 1997
J2SE	1.2	8 de diciembre de 1998
J2SE	1.3	8 de mayo de 2000
J2SE	1.4	6 de febrero de 2002
J2SE	5.0	30 de septiembre de 2004
Java SE	6	11 de diciembre de 2006
Java SE	7	julio de 2011
Java SE	8	marzo de 2014
Java SE	9	en difusión

Tabla 3 - Histórico de versiones de Java

2.3.2.3 Uso dado en la aplicación

Se trata del lenguaje de programación más utilizado actualmente y el que mayores posibilidades ofrece a la hora de encontrar tecnologías que aporten soluciones, como las que necesita este proyecto.

Además, el entorno elegido para desarrollar la aplicación, Eclipse, está implementado propiamente en Java, así como su distribución Eclipse RCP, usada para crear cada uno de los módulos de los cuales se compone la aplicación. También todos los plugins elegidos como librerías gráficas están desarrollados en Java.

2.3.2.4 JAXB

Se trata de una herramienta Java para analizar ficheros XML, disponible dentro del paquete *'javax.xml.bind'* e incluido en el JRE (1.8). Su funcionamiento se basa en la definición de una serie de clases Java que se corresponden directamente con elementos (tags) XML. Además, los atributos de las clases java podrían ser atributos de los propios elementos XML.

De esta forma, un fichero XML será convertido directamente a su equivalente de clases Java y viceversa, pudiendo volcar sobre un fichero de texto en formato XML una estructura de clases Java dada. Además, la herramienta JAXB permite generar automáticamente un fichero XSD que define la estructura XML utilizada.

En este desarrollo en concreto, se ha usado esta tecnología para definir una estructura de clases Java que se corresponda con cada uno de los posibles elementos de los ficheros de test XML a crear, así como de sus atributos. De esta manera, la conversión de código XML a su correspondiente en clase Java se realizará de forma automática, lo cual es muy útil en esta aplicación y que gracias a la implementación realizada, independiente por completo del desarrollo, podría ser reutilizada en cualquier otro entorno Java que quisiera hacer uso de este tipo de ficheros XML.

2.3.3 Eclipse

Figura 7 - Icono de Eclipse
(Fuente: https://eclipse.org/)

Se trata de la plataforma de desarrollo Java más utilizada, diseñada para ser extendida de forma indefinida a través de plugins. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene en mente un lenguaje específico, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java, usando el **plugin JDT** que viene incluido en la distribución estándar del IDE. [18]

Proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones. Compañías como IBM o Google utilizan el marco de Eclipse en gran medida y, por tanto, aseguran que sea flexible, rápido y en constante evolución. Además, potencia la existencia de una gran comunidad de individuos que proporcionan apoyo, información y extensiones. [19]

La definición que da el proyecto **Eclipse** acerca de su software es: [20]

"una especie de herramienta universal, un IDE abierto y extensible para todo y nada en particular".

Mientras que la definición de **plugin** sería: [21]

"un 'plugin' es una colección de archivos y un fichero de configuración (MANIFEST.MF) que describe dicho complemento y sus dependencias."

2.3.3.1 Características

Las principales características son: [18]

- *Perspectivas, editores y vistas*: en Eclipse el concepto de trabajo está basado en las perspectivas, que no es otra cosa que una pre-configuración de ventanas y editores, relacionadas entre sí, y que nos permiten trabajar en un determinado entorno de trabajo de forma óptima.

- *Gestión de proyectos*: el desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios,… El IDE nos proporcionará asistentes y ayudas para la creación de proyectos. Por ejemplo, cuando creamos uno, se abre la perspectiva adecuada al tipo de proyecto que estemos creando, con la colección de vistas, editores y ventanas pre-configurada por defecto.

- *Depurador de código*: se incluye un potente depurador, de uso fácil e intuitivo, y que visualmente nos ayuda a mejorar nuestro código. Para ello sólo debemos ejecutar el programa en modo depuración (con un simple botón). De nuevo, tenemos una perspectiva específica para la depuración de código, la perspectiva depuración, donde se muestra de forma ordenada toda la información necesaria para realizar dicha tarea.

- *Extensa colección de plugins*: están disponibles en una gran cantidad, unos publicados por Eclipse, otros por terceros. Al haber sido un estándar de facto durante tanto tiempo (no el único estándar, pero sí uno de ellos), la colección disponible es muy grande. Los hay gratuitos, de pago, bajo distintas licencias, pero casi para cualquier cosa que nos imaginemos tenemos el *plugin* adecuado.

2.3.3.2 Versiones

En cuanto a sus versiones, existe una clara diferencia entre las versiones antiguas, denominadas 3.x, con respecto a las nuevas, denominadas 4.x. En la actualidad, el desarrollo con el modelo de programación de Eclipse 3.x es mucho

más estable y tiene mejor apoyo de las herramientas que el de Eclipse 4. Por otra parte, Eclipse 4.x es mucho más flexible y simple de utilizar.

A continuación se muestra una tabla con los nombres de las diferentes versiones publicadas, así como su fecha de lanzamiento. [20]

Versión	Versión de plataforma	Fecha de lanzamiento
Eclipse 3.0	3.0	28 de junio de 2004
Eclipse 3.1	3.1	28 de junio de 2005
Callisto	3.2	30 de junio de 2006
Europa	3.3	29 de junio de 2007
Ganymede	3.4	25 de junio de 2008
Galileo	3.5	24 de junio de 2009
Helios	3.6	23 de junio de 2010
Indigo	3.7	22 de junio de 2011
Juno	4.2	27 de junio de 2012
Kepler	4.3	26 de junio de 2013
Luna	4.4	25 de junio de 2014
Mars	4.5	24 de junio de 2015
Neon	4.6	Junio de 2016

Tabla 4 - Histórico de versiones de Eclipse

Este proyecto se ha desarrollado usando la versión Eclipse 4.3 (Kepler), que se trata de la versión más estable cuando se ha iniciado el desarrollo de la aplicación. Y no solo la estabilidad de la versión ha sido importante a la hora de elegir una u otra, sino que se ha determinado así es la versión más reciente que acepta correctamente las últimas versiones de los demás plugins utilizados.

2.3.3.3 Distribuciones

La siguiente ilustración muestra algunas de las diferentes distribuciones disponibles, directamente obtenidas de la web de Eclipse:

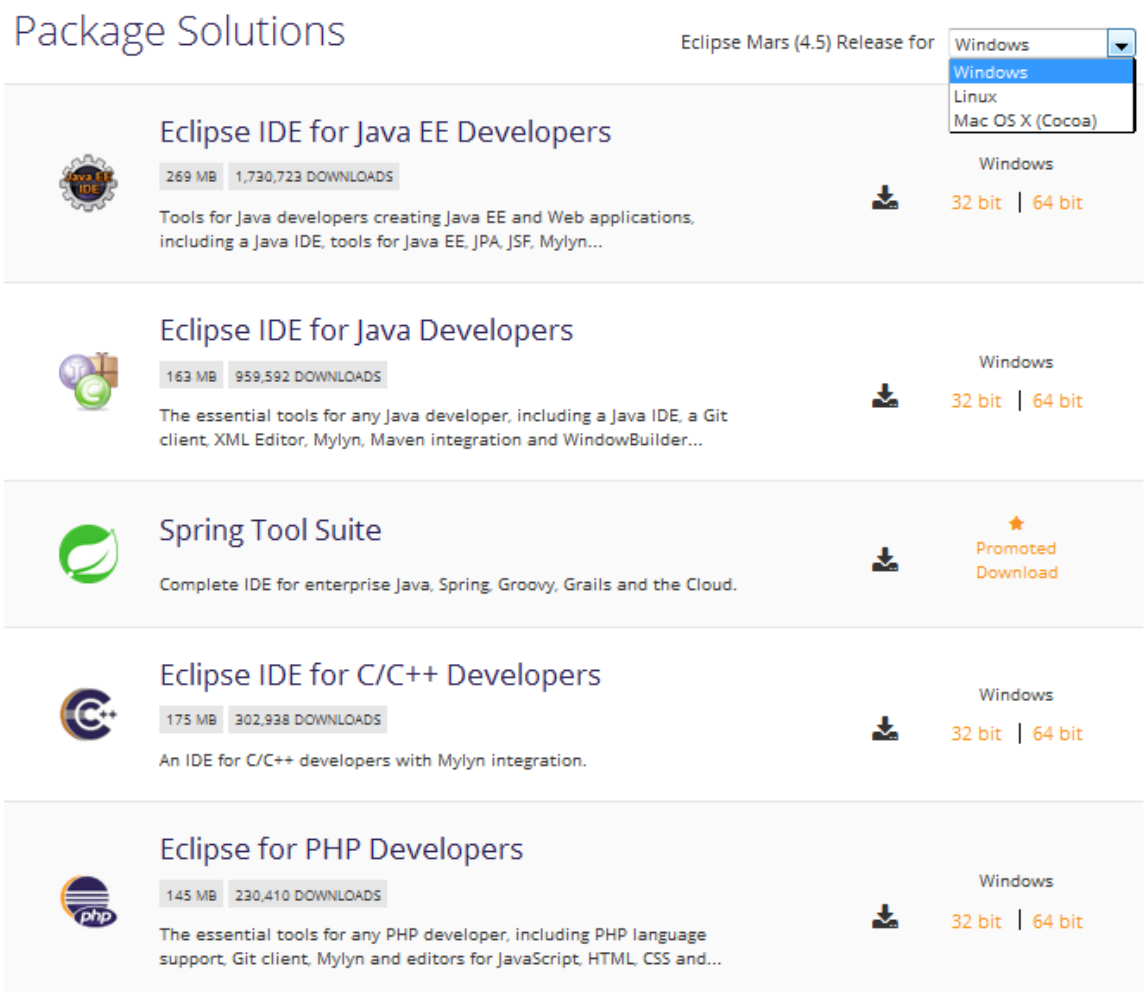


Figura 8 - Algunas distribuciones de Eclipse
(Fuente: https://www.eclipse.org/downloads/)

La elegida para desarrollar este proyecto ha sido la que contiene los plugins relativos al desarrollo RCP (Rich Client Platform) (y RAP (Remote Application Platform)):

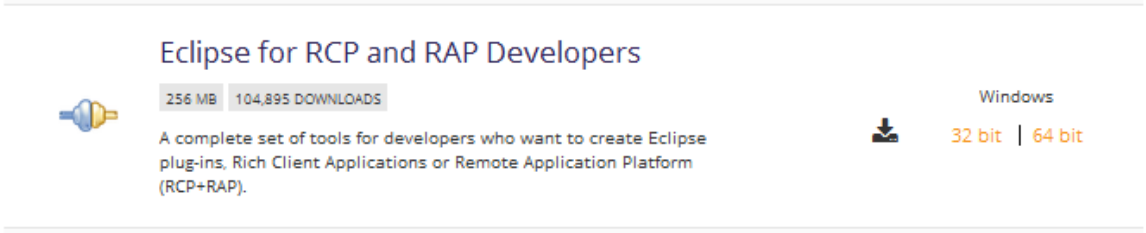


Figura 9 - Instalación de Eclipse RCP
(Fuente: https://www.eclipse.org/downloads/)

La elección de ésta y no otra distribución tiene que ver con que Eclipse RCP ya contiene por defecto los plugins y demás extensiones que se van a necesitar a

la hora de implementar la aplicación. Esta versión será la que permita crear nuevos proyectos de plugin, que contengan la estructura y las dependencias necesarias a la hora de generar la interfaz gráfica, con sus vistas, opciones, perspectivas, editores, etc.

2.3.4 Plugins de Eclipse

En este apartado se cita una serie de plugins utilizados en el desarrollo del proyecto, desde el básico que permite usar el editor de Java, pasando por los plugins que proporcionan herramientas para la creación de la interfaz gráfica, hasta llegar a plugins que han permitido generar un código legible y estructurado. Además, se hablará de la internacionalización de la aplicación y de los plugins incluidos para dar soporte a una de las características principales de este proyecto, como es la posibilidad de configurar repositorios de contenido que permitan generar un control de versiones de los proyectos y ficheros de test generados por la aplicación.

2.3.4.1 Eclipse JDT

Se trata del plugin encargado del soporte del IDE al lenguaje Java, incluido en la versión estándar de Eclipse por defecto. Es en sí mismo el editor utilizado para desarrollar la aplicación.

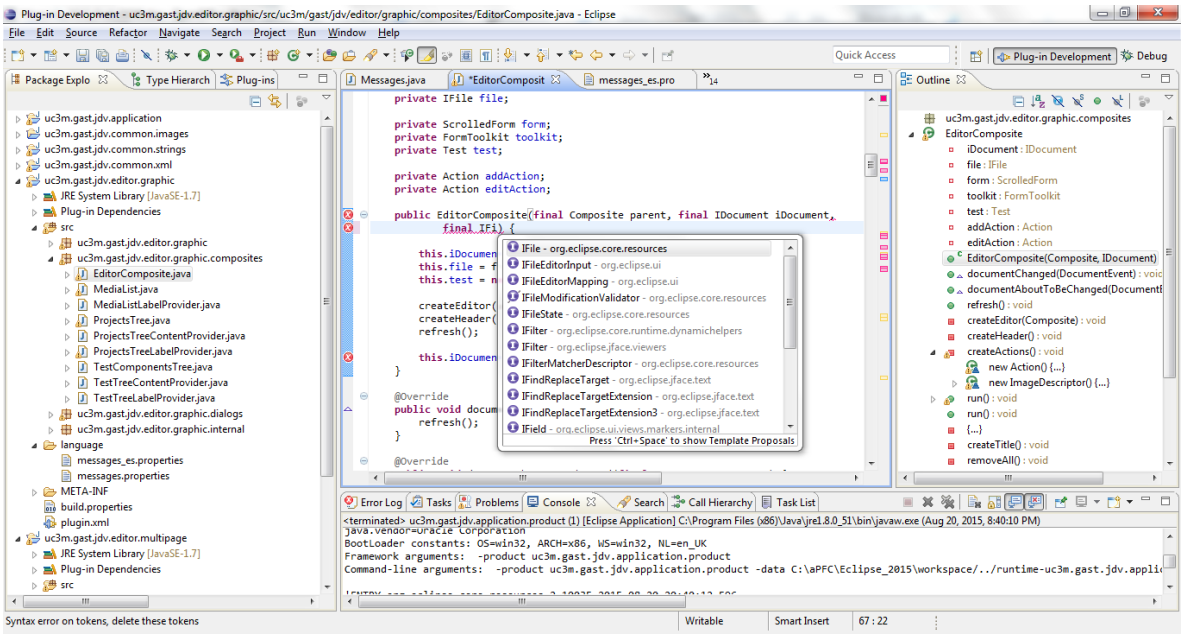


Figura 10 - Eclipse JDT (IDE)

Cuando se visualiza un proyecto Java, se abre la perspectiva correspondiente. Está formada por dos vistas: Outline y Package Explorer. La vista **Outline** se encarga de mostrar el esquema de la clase abierta en el editor activo. Una cuestión a tener en cuenta, es que cuando se tiene una vista activa, se

visualizan en la barra de herramientas opciones a modo de iconos extra, que permitirán el acceso rápido a las funciones más usadas de dicha vista. [18]

El coloreado de código en el editor es una característica muy interesante, realizando para ello el reconocimiento sintáctico de todas aquellas palabras que son reservadas en el lenguaje Java. Asimismo permite completar el código automáticamente (code completion), con sugerencias dependientes del contexto, lo cual permitirá escribir el código más rápidamente.

Se podrá configurar el formateo de código, la forma de escribir los comentarios, incluyendo comentarios para la posterior creación del Javadoc. Es posible generar los esqueletos de clase automáticamente, generación de métodos de obtención y asignación de valores a variables de manera automática, y un largo etcétera de funcionalidades.

2.3.4.2 Eclipse RCP

Se trata de un framework a partir del cual se construyen aplicaciones sobre la base de Eclipse. Es decir, nos proporcionan un Eclipse "vacío" que es posible modificar para crear aplicaciones de gestión o cualquier aplicación de escritorio que se desee. Una de sus mayores ventajas en que permite reutilizar el sistema de ayuda de Eclipse, el sistema de perspectivas y vistas, los comandos y opciones, editores, y demás funcionalidades.

Podría decirse que es un entorno completo que maneja el ciclo de vida de una aplicación de escritorio. Esto es importante, ya que provee de mecanismos para controlar aspectos críticos como la interfaz de usuario, la configuración, la modularidad, la actualización y la distribución, dejando espacio para concentrar los esfuerzos de desarrollo en lo realmente importante de la aplicación, y no en aspectos "triviales" que generalmente consumen gran parte del tiempo a la hora de desarrollar cualquier aplicación. [22]

En otras palabras, es una tecnología que se adecua al conjunto de componentes y módulos que más conviene, y así acoplarlos a la funcionalidad particular para lograr construir una aplicación de escritorio completa en una fracción del tiempo estimado, con menor índice de defectos, con interfaces de usuario consistentes e independiente de la plataforma operativa (en la medida de lo posible). [22]

Una característica importante de una plataforma RCP es su arquitectura, ya que es ella la que dicta las normas de construcción y acoplamiento de los plugins.

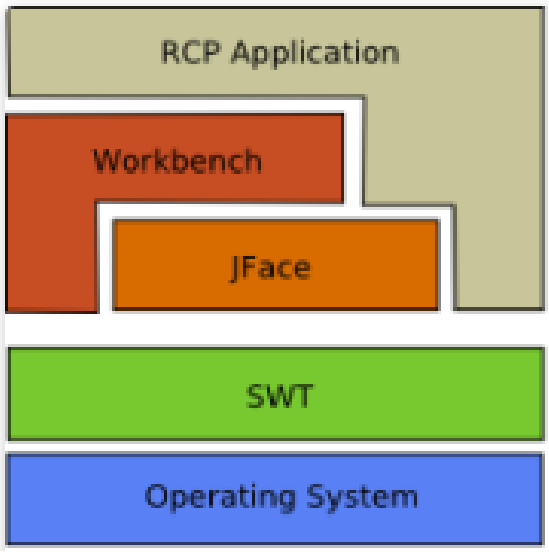


Figura 11 - Arquitectura de Eclipse RCP
(Fuente: http://www.infoq.com/articles/eclipse-rap-casestudy/)

2.3.4.2.1 Estructura de un Plugin

Dado que Eclipse RCP se basa precisamente en eso, en la definición de plugins, se va a definir qué ficheros y carpetas componen un proyecto de este tipo: [23]

- *META_INF/MANIFEST.MF:* archivo que define el nombre del plugin, la versión, quién lo desarrolló y el identificador del mismo. Indicará a Eclipse si se trata o no de un plugin y qué clase debe ejecutar para inicializarlo. Permite definir a qué (Extension Points de otros) plugins se conecta o se tiene dependencia, y también que funcionalidades o paquetes Java se exportan.

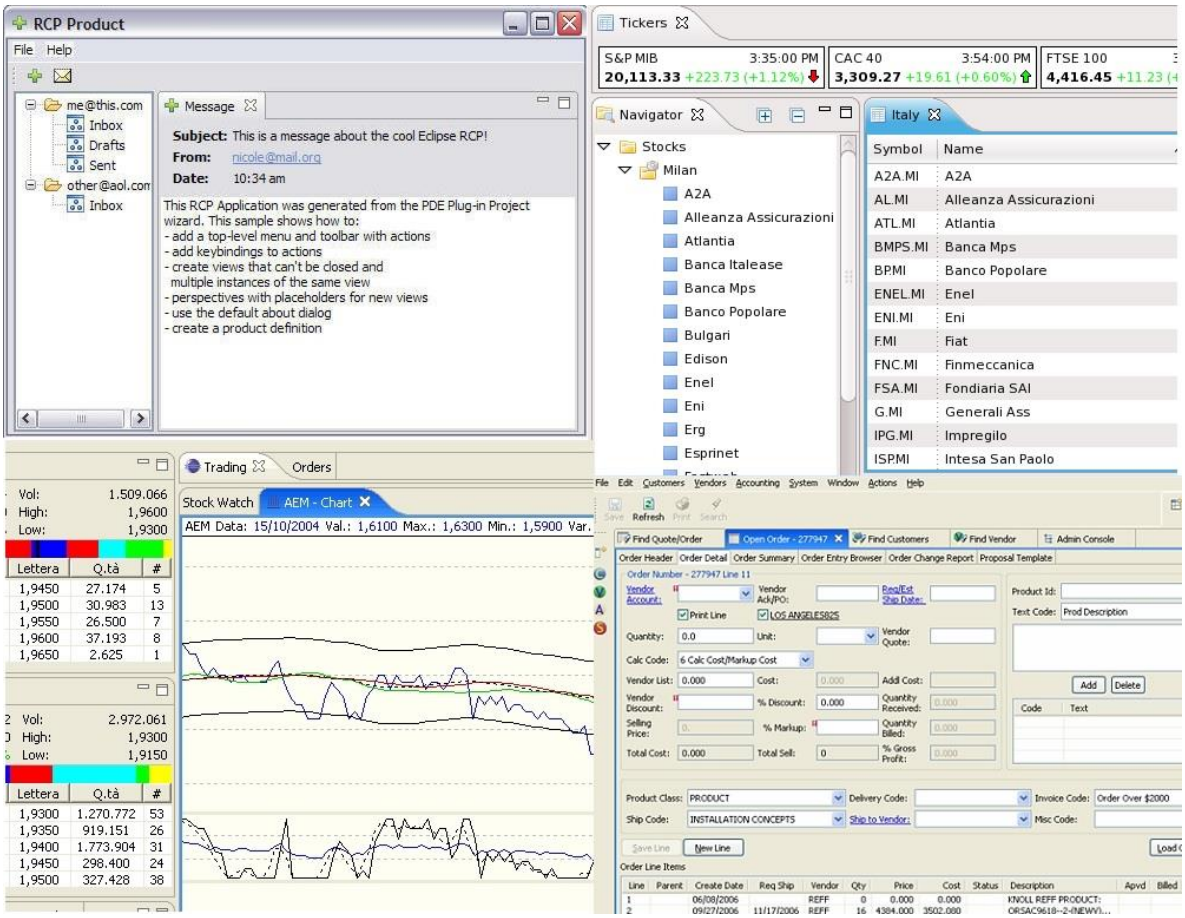
- *plugins.xml:* define como otros plugins se conectarán a este (Extension Points) y las extensiones de otros de las que se hará uso.

- *build.properties:* es el fichero en el cual se definen los recursos a incluir a la hora de generar un producto donde se incluya dicho plugin.

- *src:* carpeta en la cual se colocan las clases Java, organizadas en paquetes. Se trata de la implementación del código del plugin en sí.

- Cualquier otro tipo de recurso, como imágenes, ficheros de configuración, etc., se podrán incluir en cualquier otra carpeta creada en la raíz del proyecto de plugin.

En el anexo 8.4 se muestra cómo se crea una aplicación RCP a partir de la definición de un plugin.



2.3.4.2.2 *Ejemplos de aplicaciones RCP*

A continuación se muestran capturas de algunos ejemplos de aplicaciones realizadas mediante Eclipse RCP, con la utilización de numerosos plugins que se añaden a uno que forma la aplicación RCP y que tenga dependencia con todos los demás.

Figura 12 - Ejemplos de implementaciones sobre Eclipse RCP

(Fuentes: http://www.ibm.com/developerworks/lotus/library/expeditor-eclipse/, http://www.eclipsetrader.org/, https://eclipse.org/community/rcpos.php, http://sanghoon2.tistory.com,)

2.3.4.2.3 *Uso dado en la aplicación*

Esta estructura de plugins es la misma que se va a usar en la aplicación, definiendo una serie de plugins, cada uno con su funcionalidad específica y creando alguna dependencia entre ellos para generar el sistema completo.

De este modo se ha conseguido separar el desarrollo en partes que por sí solas tengan sentido, y que posteriormente puedan ser reutilizadas en otros sistemas formados por otro plugins. De esta forma se crea una estructura de funcionalidades independientes que se complementen las unas a las otras, para una vez todas juntas lograr dar sentido y funcionalidad a la aplicación.

2.3.4.3 SWT

Se trata de un conjunto de herramientas gráficas para el lenguaje de programación Java. Originalmente fue desarrollado por IBM y ahora parte de la Fundación Eclipse. Es una alternativa a Swing y JavaFX. [24]

Como su propio nombre indica, se trata de un conjunto de controles, denominados *widgets*, y cuya definición podría ser: [25]

"un widget es un componente visual o control básico, reutilizable, que permite en una aplicación gráfica interaccionar con él y así controlar los datos que en esta se muestran"

No debe confundirse con los widgets de escritorio, que son pequeñas aplicaciones que proveen información visual al usuario y que permite ser fácilmente accedida: relojes, calendarios, calculadoras, notas y demás programas de escritorio. [25]

SWT utiliza las API de la interfaz gráfica de usuario nativas como la API de Windows o GTK para crear sus widgets a través de JNI. Recupera la idea original de la biblioteca AWT de utilizar componentes nativos, con lo que adopta un estilo más consistente en todas las plataformas, pero evita caer en las limitaciones de ésta. La biblioteca Swing, por otro lado, está codificada enteramente en Java, pero frecuentemente se le acusa de no brindar una experiencia idéntica a la de una aplicación nativa. [26]

Sin embargo, el precio a pagar por esta mejora es la dependencia (a nivel de aspecto visual y no de interfaz de programación) de la aplicación resultante del sistema operativo sobre el cual se ejecuta. [26]

Además, la interfaz del workbench de Eclipse también depende de una capa intermedia de interfaz gráfica de usuario (GUI) llamada **JFace**, que simplifica la construcción de aplicaciones basadas en SWT. [26]

Estos widgets no tienen mucha diferencia con los que proporcionan las librerías de Swing o AWT, sino que los mejoran y los particularizan para ser usados dentro del entorno de Eclipse.

Por todas estas razones se ha decidido usar esta tecnología, ya que proporciona multitud de componentes gráficos y opciones, de forma directa o indirectamente a través de otros plugins que lo usan como base. Además, al

tratarse de la base sobre la cual se construyen las aplicaciones gráficas de RCP, su uso está incluido indirectamente.

2.3.4.4 JFace

Se trata de un conjunto de widgets, construidos sobre SWT, que permite realizar interfaces de usuario más complejas. Fue desarrollado por IBM para facilitar la construcción del entorno de desarrollo Eclipse, pero su uso no está limitado a éste. [27]

Proporciona una serie de herramientas muy frecuentes a la hora de desarrollar interfaces gráficas de usuario, tales como cuadros de diálogo, listas, tablas, árboles, etc. que evitan al programador la tediosa tarea de lidiar manualmente con los widgets de SWT.

Además, sobre muchos de estos componentes se puede usar un modelo de datos que mejorará el rendimiento de estos a la hora de ser usados, sobre todo cuando se dispone de un número elevado de datos a ser representados. Junto a esto, otras de las funciones que nos proporcionan muchos de los componentes de JFace, es la de tener proveedores (llamados 'providers') sobre sus tipos de datos que permite que la información que se muestra sea fácilmente modificada, tanto a la hora de mostrar sus valores, como de cambiar su aspecto, ya sea la fuente, los colores o las imágenes que acompañan a los datos.

Su objetivo principal es liberar al desarrollador, para que no tenga que preocuparse por el sistema de widgets subyacente o la solución de problemas que son comunes en casi todas las aplicaciones de interfaz de usuario.

Una de las principales preocupaciones del grupo Eclipse al desarrollar JFace fue que en ningún caso quieren ocultar la implementación del componente SWT del programador. JFace es completamente dependiente de SWT, pero SWT no depende de JFace. Es decir, el Workbench de Eclipse se basa tanto JFace como en SWT, y en algunos casos no es necesario pasar por JFace y se accede directamente a SWT.

2.3.4.4.1 Paquetes JFace

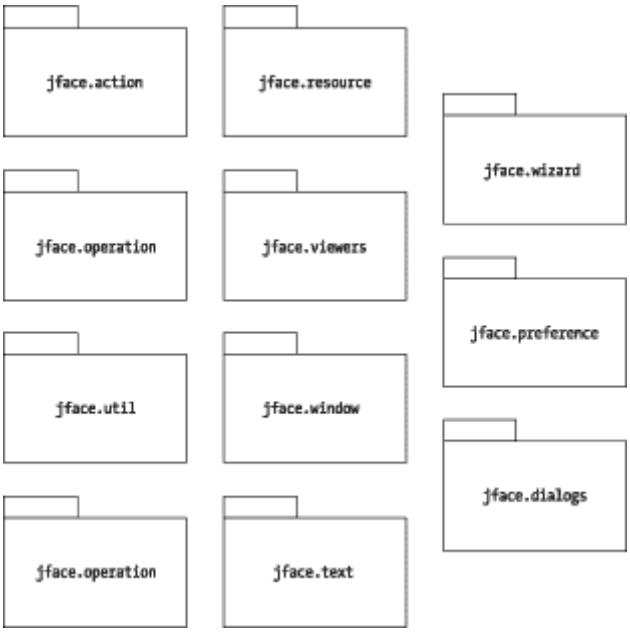


Figura 13: paquetes de la librería JFace
(Fuente: http://www.javaworld.com/article/2074837/core-java/rich-clients-with-the-swt-and-jface.html)

La API de JFace se compone de una sería de paquetes, de los cuales cabe destacar: [28]

- **Window**: El paquete *org.eclipse.jface.window* ofrece la creación de las ventanas y facilidades a la hora de manejar la aplicación. De particular interés es la clase *ApplicationWindow*, que proporciona una ventana de aplicación de nivel superior y encapsula el bucle de eventos SWT.

- **Viewers**: El paquete *org.eclipse.jface.viewers* proporciona un marco de elementos complejos basados en listas, tablas y árboles, como son la *ListViewer*, *TableViewer* y *TreeViewer*, que permiten tener un modelo vista-controlador (MVC) sobre este tipo de widgets SWT y de esta forma adaptar su contenido a un modelo de datos determinado. La presentación y el funcionamiento (performance) adecuado forman parte también de estos elementos.

- **Dialogs**: El paquete *org.eclipse.jface.dialogs* ofrece varios cuadros de diálogo de uso común. Además, permite extender muchos de estos diálogos.

- **Actions**: El paquete *org.eclipse.jface.actions* proporciona la ayuda necesaria a la hora de definir acciones comunes, que van a ser usadas en

diferentes lugares de la aplicación. Es decir, permite compartir el comportamiento entre dos o más componentes de la interfaz de usuario, como por ejemplo un elemento de menú y el botón de la barra de herramientas.

- **Wizards**: El paquete *org.eclipse.jface.wizard* proporciona un marco azanzado para crear wizards, que no son más que familia de cuadros de diálogo que automatizan tareas repetitivas y complejas. Es en sí un diálogo con más de una pantalla.

- **Resource**: El paquete *org.eclipse.jface.resource* proporciona soporte a la gestión de los recursos, como fuentes SWT e imágenes.

- **Text**: El paquete *org.eclipse.jface.text* y sus sub-paquetes proporcionan un marco para la creación, manipulación, visualización y edición de documentos de texto.

2.3.4.4.2 Ejemplos

A continuación se muestran unos ejemplos de componentes realizados mediante JFace, como son tablas, árboles o diálogos:

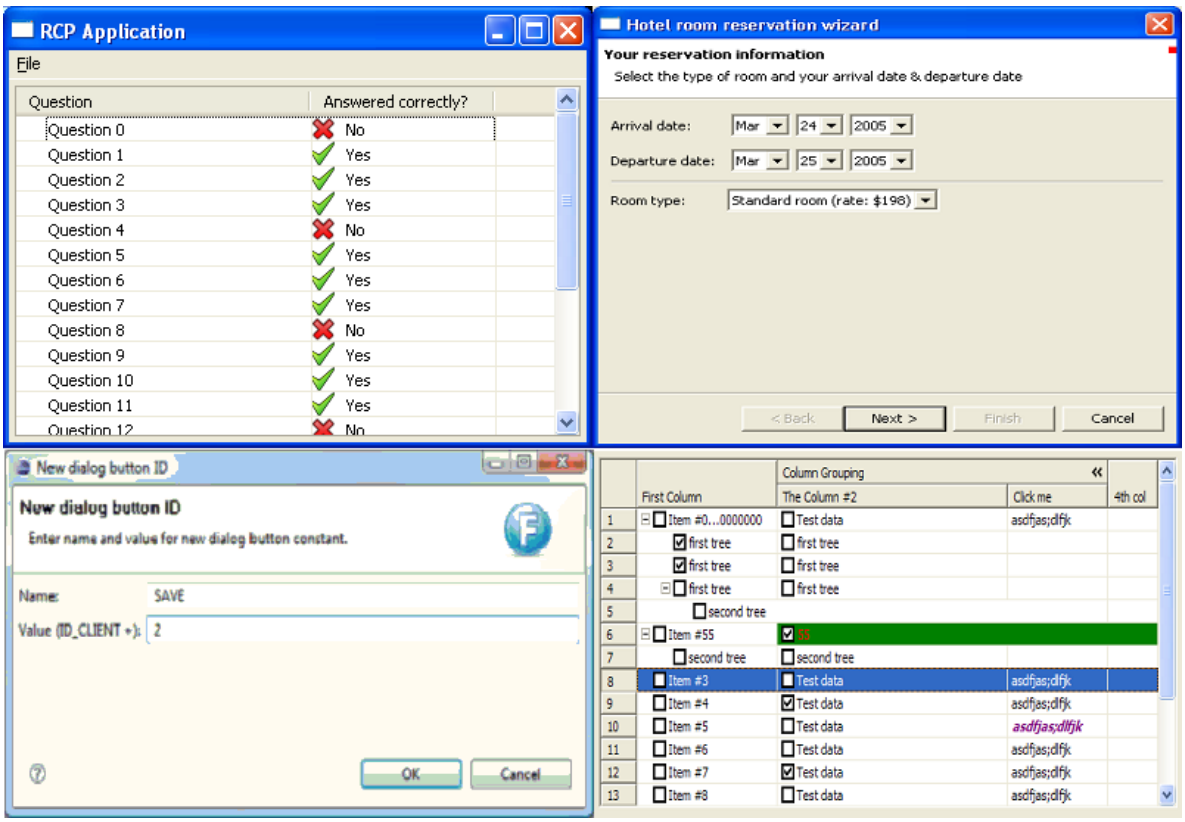


Figura 14: Ejemplos de componentes JFace

(Fuentes: https://tsoueid.wordpress.com/2009/07/31/103/,
http://www.java2s.com/Code/Java/SWT-JFace-Eclipse/WizardDemo.htm,
http://help.eclipse.org/mars/index.jsp,
http://stackoverflow.com/questions/529211/jface-tableviewer-cell-span)

2.3.4.4.3 Uso dado en la aplicación

Como se ha descrito en este punto, JFace es una capa superior a SWT que proporciona funcionalidades extras a la hora de implementar diálogos, árboles y tablas, y este es precisamente el uso que se le quiere dar en la aplicación.

Se reutilizan funcionalidades propias de Eclipse basadas en JFace, a la hora de definir diálogos por ejemplo, y también se crean nuevas tablas, árboles y diálogos para dotar al sistema de un mayor número de opciones.

2.3.4.5 Eclipse Forms

Eclipse Forms es un plugin que proporciona el apoyo para la creación de interfaces de usuario de estilo web, portable en todas las distribuciones de Eclipse UI. Dicha funcionalidad está contenida en el plugin "*org.eclipse.ui.forms*", es opcional dentro de Eclipse RCP y solo está disponible para las versiones 3.x de Eclipse. [29]

Eclipse Forms está basado en SWT y JFace, pero no fue diseñado para competir contra ellos sino para ofrecer una alternativa diferente. Se compone de una serie de widgets personalizados y elegidos cuidadosamente, diseñados y basados en unas clases que tratan de conseguir y dar un efecto Web a vistas o editores de Eclipse. [29]

El hecho de que cada parte del formulario sea accesible en todo momento desde cualquier lugar, hace de estos recursos una herramienta potente y atractiva. El logro de que la misma sea tan flexible, requiere un amplio apoyo DOM y la interacción continua con las diferentes propiedades. [29]

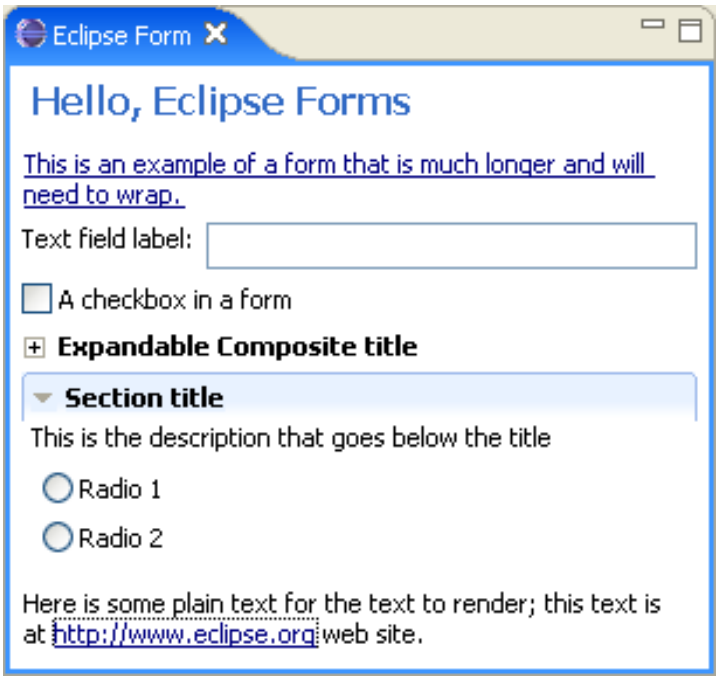


Figura 15: Ejemplo de Eclipse Forms
(Fuente: https://eclipse.org/articles/Article-Forms/article.html)

Los aspectos a tener en cuenta son: [29]

- La clase *FormToolkit* sirve como una factoría para la creación de los elementos de la interfaz de usuario necesarios. Esta factoría ajusta la apariencia de los elementos estándar SWT y JFace a la API de Form. Además, otro tipo de elementos de interfaz de usuario existentes se pueden asemejar a la forma de esta API a través del método de un método de adaptación.

- Se tiene una única instancia de *Form* o *ScrolledForm*, proporcionada por la factoría *FormToolkit*. Están compuestos de una cabecera, una barra de herramientas y el contenedor donde colocar los demás elementos gráficos. Este elemento se puede usar dentro de instancias tales como vistas o editores de Eclipse.

- Posee un juego de herramientas para gestionar colores, grupos de hipervínculo y otros aspectos del formulario

- Dispone de un controlador para la distribución y colocación de los controles de una manera similar a un algoritmo de diseño de tabla HTML.

- Añade un conjunto de controles personalizados diseñados específicamente para ser usados en un Form (hiperenlace, hipervínculo imagen, contenedor desplazable, sección…).

- Permite tener un editor de varias páginas donde la mayor parte o la totalidad de las páginas son Forms.

Aunque nada en el diseño de Eclipse Forms le impide la creación de un formulario en un cuadro de diálogo, la mayoría de los escenarios de uso se han centrado en el uso de las formas en las vistas y editores, en lugar de diálogo y asistentes. Es una cuestión de coherencia, pues los diálogos o wizards basadas en Forms se ven muy extraños, cuando todos los demás cuadros de diálogo son "normales". Sin embargo, es una zona interesante para explorar en el futuro. [29]

En esta aplicación concretamente, el uso se basa en la representación de los ficheros de test dentro de la vista de pre-visualización incluida en el editor XML. Ello permite dar un enfoque a modo de web de cómo se representarían los ficheros de test, utilizando las propiedades de Eclipse Forms a la hora de definir un título, crear toolbars con opciones, crear secciones que puedan ser replegadas o desplegadas, etc.

2.3.4.6 Control de versiones: CVS y Subversion

Son dos de las herramientas o sistemas más utilizados a la hora de gestionar el control de versiones de directorios y ficheros, tanto Java como de otras tecnologías. Se puede afirmar que las instancias de CVS y Subversion son en sí repositorios, que funcionan como un almacén de ficheros, con sus versiones y su historial de cambios. [30]

A grandes rasgos permiten: [30]

- que varias personas puedan trabajar en un mismo proyecto sin afectarse mutuamente en el trabajo que realiza,
- que haya un histórico de revisiones, por seguridad y para poder recuperar una versión anterior de nuestro código,
- gestionar versiones de la aplicación, derivaciones y demás,
- realizar copias de seguridad de la información almacenada.

Subversion se trata de una evolución natural de CVS, mejorando algunas de las carencias de este último. La siguiente tabla muestra cuáles son sus características, permitiendo observar que diferencias existen entre ambas. [30]

CVS	Subversion
Licencia gratuita	Licencia gratuita

Multiplataforma	Multiplataforma
Orientado a ficheros: versiona los ficheros, no el proyecto. Esto quiere decir que cada modificación en cada fichero hace variar la versión del fichero modificado.	Orientado a proyectos: versiona los proyectos, no los ficheros. Es una opción mucho más intuitiva que no la de versionar ficheros.
Envía ficheros completos: cuando se hace una modificación en un fichero, se sube al repositorio el fichero entero, en vez de sólo los cambios.	Envía sólo cambios: cuando se hace una modificación en un fichero, se sube al repositorio el cambio realizado, no el fichero entero. Más eficiencia.
Soporte Unicode limitado: puede traer problemas con caracteres "raros".	Soporte Unicode: en principio no debería dar problemas de esta índole.
Renombrar/eliminar: no da soporte. Hay que hacerlo manualmente: copiar con otro nombre y borrar el antiguo.	Renombrar/eliminar: da soporte parcial. El usuario lo hace en una única operación pero SVN internamente lo hace las dos cosas en una transacción de forma transparente al usuario: copiar con otro nombre y borrar el antiguo.

Tabla 5 – Comparativa CVS contra Subversion

(Fuente: http://www.iiia.csic.es/udt/es/blog/jrodriguez/2008/subversion-vs-cvs-guia-rapida-subversion-svn)

El desarrollo efectuado en la aplicación para configurar el uso de estas dos tecnologías ha consistido en añadir los plugins de Eclipse definidos por ambas. De esta forma se han importado dichas funcionalidades y de manera automática se añadirán a las opciones disponibles en la aplicación.

La utilidad que ambas tecnologías aportan a la aplicación será la de poder compartir los proyectos de test generados (o cualquiera de los recursos que lo componen) o descargarse otros proyectos desde cualquier repositorio. Y además lo más importante, disponer de un control de versiones de cada uno de los ficheros de test generados o de cualquier otro recurso. También la posibilidad de trabajar en grupo será un punto importante.

2.3.4.7 Calidad del código

Uno de los requisitos a la hora de desarrollar el código que compone la aplicación es dotarle de claridad, estructura, limpieza, orden, etc., ya que no solo se quiere separar la funcionalidad en diferentes plugins, dependiendo de su

función, sino que además se busca que el código en sí mismo cumpla unas reglas de codificación que le doten de calidad.

Para ello se emplean diferentes herramientas proporcionadas por Eclipse, como son el 'formatter', el 'clean up' y las funcionalidades que añade el plugin 'CheckStyle'.

De ello se habla a continuación y además se añade una guía de cómo configurar estas opciones en el anexo 8.5.

2.3.4.7.1 Clean-up / Formatter

Aunque una aplicación funcione perfectamente, el esfuerzo de mantenimiento del código que compone ésta puede ser más costoso (lo que implica inversión de tiempo y dinero) que rehacer la aplicación en su totalidad. Este esfuerzo está directamente relacionado con varios factores, y uno de ellos es la claridad, limpieza y formato del código fuente.

Eclipse incorporada opciones que permiten realizar la limpieza y el formato de código de forma automática. Estas funciones ayudan a mantener un código limpio, bien organizado, con una tasa de comentarios razonable, una tabulación adecuada de las líneas de código, que dicho código cumpla con las mínimas exigencias que la programación Java requiere, etc. En definitiva, ayuda a escribir el código y lo hace de forma estructurada, cumpliendo con las reglas de codificación. [1]

La opción de '**Clean-up**' que ofrece Eclipse se encarga de hacer una limpieza de código y de completar aquellas líneas de código que sean necesarias. Por tanto, se encarga de:

- añadir anotaciones que falten,
- elimina las importaciones no utilizadas,
- declara variables como final.

En definitiva, hace que el código cumpla con las reglas de codificación que dicta Java.

Sin embargo, no cambia nada del diseño del código, pues de esto se encarga el '**formatter'**. Esta opción permite:

- configurar la sangría correcta,
- gestionar el vacío de líneas,
- dar soporte al código, como pueden ser las llaves de inicio/fin,

- justificar las líneas a un máximo de caracteres,
- configurar los espacios.

2.3.4.7.2 Checkstyle Plugin

Se trata de un plugin, también conocido como *eclipse-cs*, que integra un analizador de código fuente estático en el IDE de Eclipse. Se trata de una herramienta de desarrollo de código abierto para ayudar a asegurarse de que el código Java se adhiere a un conjunto de estándares de codificación. Checkstyle hace esto mediante la inspección constante y señalando los elementos que se apartan de un conjunto definido de reglas de codificación. Así, notificará de inmediato los problemas o errores que haya en el código, de forma similar al compilador de errores o advertencias. [31]

Esto asegura a los desarrolladores observar a primera vista los posibles errores de código que se están cometiendo, permitiendo solucionarlos en el momento de la codificación. Además, dentro de un equipo de desarrollo, facilita la inclusión de unas normas comunes para los estándares de codificación (reglas de formato, longitudes de línea, etc.).

En definitiva, se trata de una herramienta muy útil a la hora de crear un código que guarde cierta calidad, unas reglas de codificación predeterminadas, y que se añade al trabajo realizado por las opciones de 'Clean-up' y 'Formatter'.

2.3.4.8 Internacionalización de la aplicación

El proceso de preparación de una aplicación para ser traducido a varios idiomas se llama internacionalización. Este término normalmente se abrevia a i18n.

Dado que esta aplicación busca ser lo más global posible, se ha incluido esta opción que permite tener un entorno en diferentes idiomas. Concretamente, es posible iniciar la aplicación en inglés (por defecto) o en español. Además, mediante la posibilidad que ofrece Eclipse de crear y añadir una 'feature' que complemente un plugin, se podría incluir cualquier otro idioma a posteriori.

En el anexo 8.4.1 se explican los pasos a la hora de internacionalizar una aplicación o un plugin de Eclipse.

2.3.5 Otras tecnologías existentes (descartadas)

A continuación se citan otras tecnologías similares a las usadas, que han sido estudiadas al inicio del proyecto como posibles soluciones, pero que finalmente fueron descartadas. Concretamente, se cita a NetBeans como posible opción en lugar de haber usado Eclipse como IDE a la hora de desarrollar el proyecto; además, se habla de Swing como posible entorno gráfico en lugar de Eclipse

RCP. Se verán las características de esta herramienta de desarrollo y de esta otra tecnología a continuación, así como sus ventajas e inconvenientes respecto a las soluciones elegidas.

2.3.5.1 NetBeans

Se trata de un entorno de desarrollo integrado, hecho principalmente para el lenguaje de programación Java. Al igual que en Eclipse, existe un número importante de plugins para extenderlo, y su principal uso en el desarrollo de aplicaciones de escritorio implementadas en Java Swing.

Sus principales características son: [32]

- Gestión de la interfaz de usuario (menús y barras de herramientas).
- Gestión de configuración de usuario.
- Gestión de almacenamiento (guardar o cargar algún tipo de dato).
- Gestión de ventana.
- Marco Asistente (soporta diálogos para a paso).
- Librería visual de Netbeans.
- Herramientas de desarrollo integrado

Además, NetBeans IDE es libre, de código abierto y multiplataforma.

Por tanto, se trata de una alternativa muy parecida a Eclipse, que perfectamente podría haber sido utilizada a la hora de desarrollar la aplicación. Sin embargo, se ha decidido usar Eclipse por multitud de razones, que se detallan en la Tabla 6 de ventajas e inconvenientes mostrada más abajo.

2.3.5.2 NetBeans RCP

Se trata de un framework para la construcción de aplicaciones de escritorio. Se encarga de automatizar buena parte de las características comunes a cualquier aplicación de escritorio, como son las preferencias del usuario, ayuda, actualizaciones automáticas, gestión de eventos, layouts, etc. [33]

Sin embargo, no ha sido tenida en cuenta debido a las facilidades que ofrece Eclipse y sus plugins a la hora de extender sus funcionalidades y de implementar un entorno de desarrollo con editores. Las ventajas e inconvenientes se observan en la tabla mostrada más abajo.

2.3.5.3 Java Swing

Se trata de una biblioteca gráfica implementada en Java, que sigue el modelo vista controlador. Incluye widgets para desarrollar interfaces gráficas de usuario, tales como cajas de texto, botones, desplegables y tablas. Sigue un simple modelo de programación por hilos, y posee las siguientes características principales: [34]

- Independencia de plataforma.
- Extensibilidad: es una arquitectura altamente particionada, donde los usuarios pueden proveer sus propias implementaciones modificadas para sobrescribir las implementaciones por defecto. Se puede extender clases existentes proveyendo alternativas de implementación para elementos esenciales.
- Personalizable: dado el modelo de representación programático del framework de swing, el control permite representar diferentes estilos de apariencia "look and feel". Además, los usuarios pueden proveer su propia implementación de apariencia, que permitirá cambios uniformes en la apariencia existente en las aplicaciones Swing sin efectuar ningún cambio al código de aplicación.

Se trata de una alternativa a los recursos gráficos que ofrece SWT, y que extienden JFace y Eclipse Forms. Sin embargo, su desarrollo es mucho más complejo y no ofrece tantas facilidades como las bibliotecas gráficas de Eclipse, de ahí que se haya desestimado su uso.

A continuación se ofrece una tabla con las ventajas e inconvenientes de cada una.

Tecnología	Ventajas	Inconvenientes
NetBeans	• herramientas Swing DnD • mejor soporte de webapps (.war, jsp y servlets) • GUI muy intuitiva • depuración sencilla • mejor soporte para PHP • permite importar proyectos de Eclipse y otros IDE's • permite programar en multitud de lenguajes • licencia GPL • mejor interfaz SVN	• más pesado, sobre todo si se tienen varios proyectos • proceso de compilación más complicado
Eclipse	• más rápido • más flexible • más plugins • más extendido • potente editor de texto • compilación en tiempo real • mejor soporte para Android • SWT, JFace, Eclipse Forms • soporte de refactorización de código	• complejidad de configuración • complejidad de uso

Tabla 6 - Ventajas e inconvenientes entre NetBeans y Eclipse
(Fuente: https://emiliosedanogijon.wordpress.com/2014/10/13/trabajo-entornos-de-desarrollo-libres-y-comercia

Tecnología	Ventajas	Inconvenientes
• **NetBeans RCP**	• es más maduro, ya que es más antiguo • plugins más fácil de comprender y usar • 100% Java • editor de widgets de Swing	• Swing proporciona peor rendimiento con respecto a SWT • carece de cierta funcionalidad de alto nivel, como las vistas y perspectivas • muy dependiente de NetBeans IDE
Eclipse RCP	• fuerte comunidad de desarrolladores • modelo de plugins más avanzado que existe • SWT, JFace y Eclipse Forms para desarrollo GUI • posibilidad de "empotrar" componentes Swing en componentes SWT • despojado de cualquier concepto relacionado con el IDE • sistema de módulos OSGi • reutilización de código • multitud de posibilidades de configuración	• dificultad a la hora de llegar a controlar todas sus posibilidades

Tabla 7 - Ventajas e inconvenientes entre NetBeans RCP y Eclipse RCP
(Fuente: http://www.javahispano.org/antiguo_javahispano_org/2004/4/17/netbeans-platform-vs-eclipse-rcp.html)

Tecnología	Ventajas	Inconvenientes
Java Swing	• es más maduro, ya que es más antiguo • 100% Java • multiplataforma • multitud de tutoriales • apoyo de OpenGL Java • muchos expertos	• peor rendimiento con respecto a SWT • componentes pesados • experiencia no idéntica a la de una aplicación nativa
SWT	• alto rendimiento • menor uso de memoria • multiplataforma • fuerte comunidad de desarrolladores • posibilidad de "empotrar" componentes Swing en componentes SWT • combina lo mejor de AWT y Swing • multitud de widgets • base de JFace y Eclipse Forms	• librerías nativas según cada sistema • 'look' nativo según SO • no es libre, sino propiedad de Eclipse

Tabla 8 - Ventajas e inconvenientes entre Java Swing y SWT
(Fuente: http://stackoverflow.com/questions/2306190/java-desktop-application-swt-vs-swi

Capítulo 3

Diseño e Implementación del Sistema

En este apartado se describe la arquitectura de la aplicación, explicar cada una de las partes que la componen y las relaciones entre ellas. De este modo se describen todos los plugins de los que se compone y sus interacciones. Se profundizará en la funcionalidad de cada plugin de forma independiente dando una explicación detallada de su funcionamiento.

3.1 Arquitectura

Una de las principales características para el desarrollo de la aplicación ha sido poder independizar los diferentes módulos del sistema mediante el uso de plugins. La aplicación está compuesta de un total de siete plugins, los cuales de detallan en la siguiente imagen además de ver las relaciones o dependencias que guardan entre ellos.

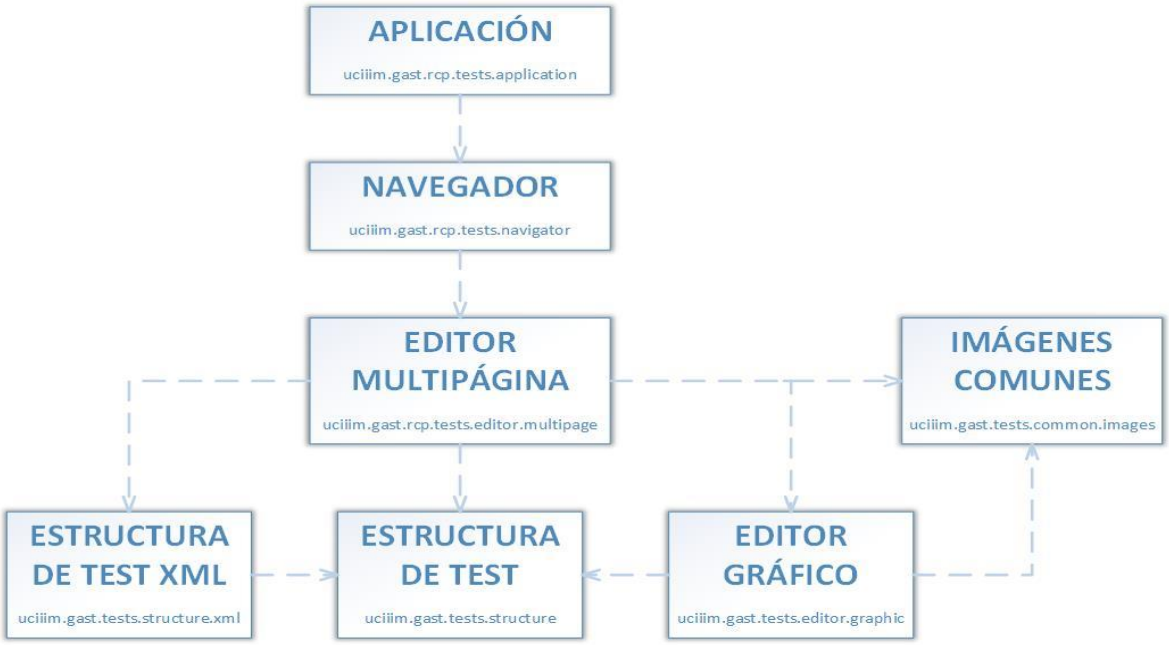


Figura 16 – Arquitectura de la aplicación

A continuación se describen las características de cada uno y las razones por las cuales se ha decido separar su funcionalidad del resto.

3.1.1 Estructura de test

Este componente no se trata de un plugin de Eclipse RCP en sí, sino que puede ser reutilizado en cualquier otro entorno Java.

Se trata de las interfaces que definen la estructura de test diseñada por este proyecto, y que a su vez es la base de cualquier aplicación de test que pueda derivarse a partir de esta. Esto es debido a que los ficheros de test, independientemente del formato que tengan, deberán cumplir la estructura que se define en estas interfaces.

De esta forma, por ejemplo, podría utilizarse en un desarrollo Android para analizar uno de los test generados mediante la aplicación que se ha definido, y así poder crear una estructura que contiene objetos Java a partir de un fichero de test dado. Esta funcionalidad es la que persigue esta aplicación, de manera que los ficheros generados puedan ser leídos y se pueda trabajar con ellos sobre cualquier otro entorno Java, siempre que sigan esta estructura e independientemente del formato en el cual estén escritos.

3.1.1.1 Estructura de clases

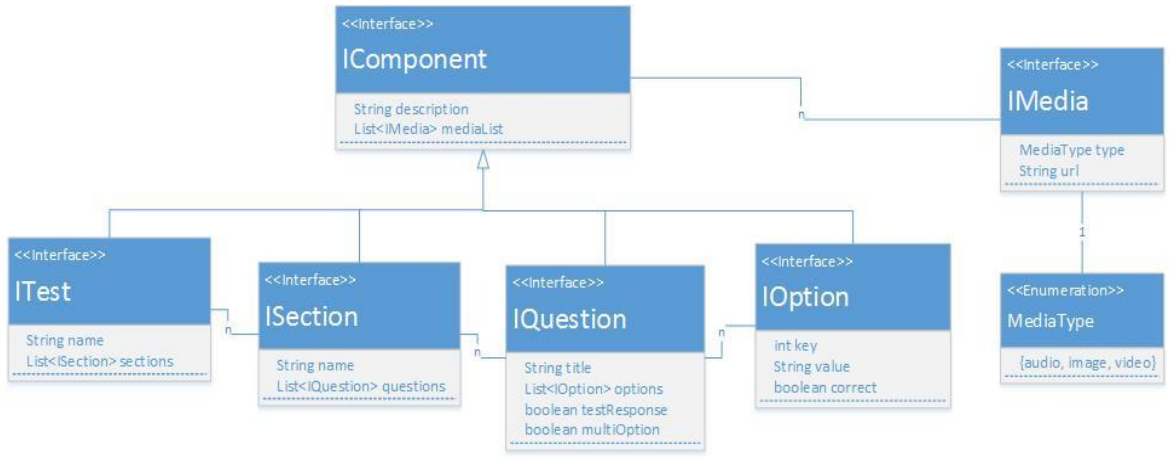


Figura 17 – Diseño de interfaces de la estructura de test

En primer lugar, describir la interfaz **IMedia**, que define la ruta a un recurso, ya sea interno al propio proyecto de test o mediante una URL a un recurso de internet. Además, tiene que declarar su tipo, en función del enumerado **MediaType**, que define si se trata de una imagen, una ruta a un recurso de audio o un video.

En segundo lugar, las interfaces que forman esta estructura son:

- **IComponent:** interfaz que define la base de los elementos de un test. El resto de interfaces que definen a un componente del test, extienden a esta. Se compone de un texto en forma de descripción de ese elemento y de una posible lista de elementos *IMedia*.

- **IOption:** es el elemento más pequeño definido en la estructura de test. Se compone de un valor en forma de texto que sería el texto de dicha opción, una bandera (flag) que indica si esta opción es o no correcta, y una clave numérica que define inequívocamente a este elemento dentro de la pregunta.

- **IQuestion:** se trata de la pregunta en sí. Está compuesta de un título, una lista de opciones de tipo *IOption*, un flag que dice si sebe incluirse o no una respuesta libre, y otro flag que define el tipo de respuesta, especificando si se trata o no de una pregunta con opciones multi-respuesta.

- **ISection:** un test se compone de una lista de secciones. Este componente almacena una lista de preguntas de tipo *IQuestion*, y define un nombre para dicha sección.

- **ITest:** se trata del elemento principal de la estructura. Engloba a todos los demás elementos creados en el test. Se compone de un conjunto de secciones de tipo *ISection*.

3.1.2 Estructura de test XML

Se trata de un plugin que implementa la funcionalidad dada por la estructura de test descrita anteriormente, de forma que los elementos que definen las interfaces van a estar representados por objetos Java que a su ver se van a corresponder con objetos XML.

De esta forma, cada uno de los componentes de tipo IComponent listados anteriormente, va a tener su implementación en este plugin, como muestra la siguiente lista con las clases de este plugin:

- **AbstractComponent**: implementa a IComponent.

- **Test**: extiende a AbstractComponent, e implementa a ITest.

- **Section**: extiende a AbstractComponent, e implementa a ISection.

- **Question**: extiende a AbstractComponent, e implementa a IQuestion.

- **Option**: extiende a AbstractComponent, e implementa a IOption.

- **Media**: implementa a IMedia.

Como puede observarse, la correspondencia entre interfaces e implementaciones es bastante lógica y sencilla. Estas clases definen mediante sus atributos los elementos que son directamente relacionados con el fichero XML, mediante el uso de JAXB, explicado en la sección 2.3.2.4.

Al igual que el plugin Estructura de Test, este tampoco se trata de un plugin propio de Eclipse RCP, por lo que podría usarse en cualquier otro entorno Java para analizar los ficheros de test creados desde esta aplicación en formato XML.

3.1.3 Imágenes comunes

Plugin común que contiene todas las imágenes e iconos usados en la aplicación, así como las clases de utilidades que permiten acceder a ellos.

Se ha tomado la decisión de centralizar este tipo de recursos, para poder modificar de manera muy sencilla uno, varios o todos los iconos de la aplicación, sin tener que hacer otro tipo de modificaciones en los demás plugins.

Respecto a las clases de utilidades, proporcionan métodos estáticos que son utilizados desde diferentes puntos de la aplicación, por lo que teniéndolo en un módulo independiente, es una buena forma de evitar la duplicidad de código.

Este plugin, al igual que los siguientes de los que se va a tratar, son plugins propios de Eclipse RCP, por lo que solo pueden usarse en un entorno de este tipo.

3.1.4 Editor gráfico

Se trata de un plugin que pre-visualiza un determinado test. Es decir, recibe como entrada un objeto de tipo **ITest** y lo transforma en un componente gráfico compuesto de secciones, preguntas y opciones de respuesta, todos ellos con sus textos, descripciones, elementos de tipo **IMedia** (imágenes, audios o videos), etc… No está directamente relacionado con el plugin que define la estructura XML, ni tiene porque formar parte del editor de ficheros multi-página que se describe a continuación, ya que como decimos, lo que recibe como entrada es un elemento de tipo **ITest**.

Es parte fundamental de este proyecto, ya que es de dónde nació la idea de tener un entorno gráfico en el cual poder crear de forma interactiva ficheros de test y poder pre-visualizar como quedaría éste de cara al usuario, a modo ejemplo.

Para ello, utiliza gran parte de las herramientas descritas en apartados anteriores, como son SWT, JFace, Eclipse Forms o la internacionalización de plugins:

- mediante **Eclipse Forms**, se ha creado el componente gráfico general del editor, dentro del cual se crea: el título y la descripción del test, todas las secciones definidas, las preguntas que dentro de las distintas secciones se van encontrando, así como las diferentes opciones a la hora de la respuesta. Además, permite añadir barras de herramientas dentro del propio editor, haciendo así más intuitivo si cabe la creación o modificación del test de cara a los usuarios.

- **JFace** proporciona todos los diálogos de adición o edición del editor, así como las tablas y árboles que se muestran en ellos.

- Mientras que **SWT**, como base de todo, permite la creación de las etiquetas, botones, inserción de imágenes, etc. que hay dentro del editor y sus diálogos.

- Por último, el editor se ha implementado de tal forma que permite la **internacionalización** de todos sus textos, ya sean los que incluye dentro de los menús como los propios que se muestran en él y en sus diálogos.

Se trata por tanto de un plugin con código muy complejo y a la vez muy específico, compuesto por muchas clases divididas en diferentes paquetes, y dentro de las cuales se ha seguido un método de desarrollo de reutilización de código. De esta forma, por ejemplo, todos los diálogos con funcionalidad similar extienden de una clase abstracta única, que define dicha funcionalidad; las tablas y árboles se han configurado con proveedores de contenido y etiquetas funcionales, de manera que los modelos que muestran se traten de forma productiva y rápida; mientras se ha creado una implementación propia de las clases definidas en la estructura de test, siendo de esta forma independiente de la estructura definida para los ficheros XML.

3.1.5 Editor multi-página

Se trata del plugin que conecta los plugins descritos hasta ahora. Por tanto, tiene dependencias con todos ellos:

- **Estructura de Test**: es el que define la estructura Java de los test que se editan en el editor multi-página.

- **Estructura de Test XML**: lo usa para convertir el texto XML del editor a una instancia **ITest**, y vuelve a convertir esa instancia código XML.

- **Editor gráfico**: incluye la página de edición gráfica que define este plugin, junto al propio editor multi-página que éste plugin contiene. Se comunican entre ellos de forma bidireccional, pasándose la instancia del test definida en el elemento **ITest**.

- **Imágenes comunes**: utiliza alguna de las imágenes que en él se definen.

Incluye los diálogos de creación de proyectos y ficheros de test. En estos, además de introducir el nombre del fichero y elegir su ubicación, se define el título del test, se permite dar una introducción a éste, y además se da la oportunidad de seleccionar un fichero por defecto, a modo de plantilla, de una lista muy extensa y variada. De esta forma, la creación de un test desde cero se hace más rápida y eficiente.

3.1.6 Navegador

Es el plugin que define la vista dentro de la cual se visualizan los proyectos de test creados, con sus carpetas, ficheros y recursos (imágenes, audios y videos). Define también el tipo de contenido que se va a mostrar en él y que opciones se van a incluir en el menú contextual.

Tiene un papel fundamental dentro de la aplicación ya que se necesita para visualizar el estado de los proyectos de test creados en el entorno de trabajo.

Tiene como dependencia al editor multi página, y con él a todos los demás plugins descritos con anterioridad.

3.1.7 Aplicación

Se trata del plugin que define el ejecutable del proyecto. Incluye como dependencia al plugin del **Navegador**, del cual se ha hablado anteriormente, y se encarga de definir la vida de la aplicación, la ventana principal, la ventana de introducción que sale al iniciar, las perspectivas,…

Sin embargo, este módulo "no es imprescindible ni forma parte en sí" de la aplicación. Esto se debe a que el plugin que define el navegador, podría incluirse como dependencia de cualquier otra aplicación de Eclipse RCP, y su funcionamiento sería el mismo.

En este caso se utiliza además para crear el ejecutable de la aplicación.

3.2 Implementación

Esta sección define la aplicación centrándose en los puntos relevantes acerca de cómo se ha estructurado la misma, que plugins la componen, que interacción tienen entre ellos, que clases son las más importantes de cada uno, que ficheros de configuración contienen, etc.

También se describe el así como qué se debería hacer para modificar, mejorar o añadir funcionalidad a la aplicación.

En primer lugar se muestran cada uno de esos plugins, explicando sus clases, sus ficheros de configuración y sus recursos; en segundo lugar se hablará de cómo un fichero en formato XML llega a ser representado por el editor gráfico, y como este lo modifica y vuelve a enviar a su origen para que se almacene en su formato original; en tercer lugar, se hablará de cómo se puede actualizar la aplicación, sus menús y opciones; a continuación se introducirá como sería posible traducir la aplicación a un nuevo idioma; y por último se hablará de cómo se procedería a la hora de reusar los componentes implementados.

3.2.1 Plugins definidos

Muestra la estructura interna de cada uno de los plugins generados.

3.2.1.1 Aplicación

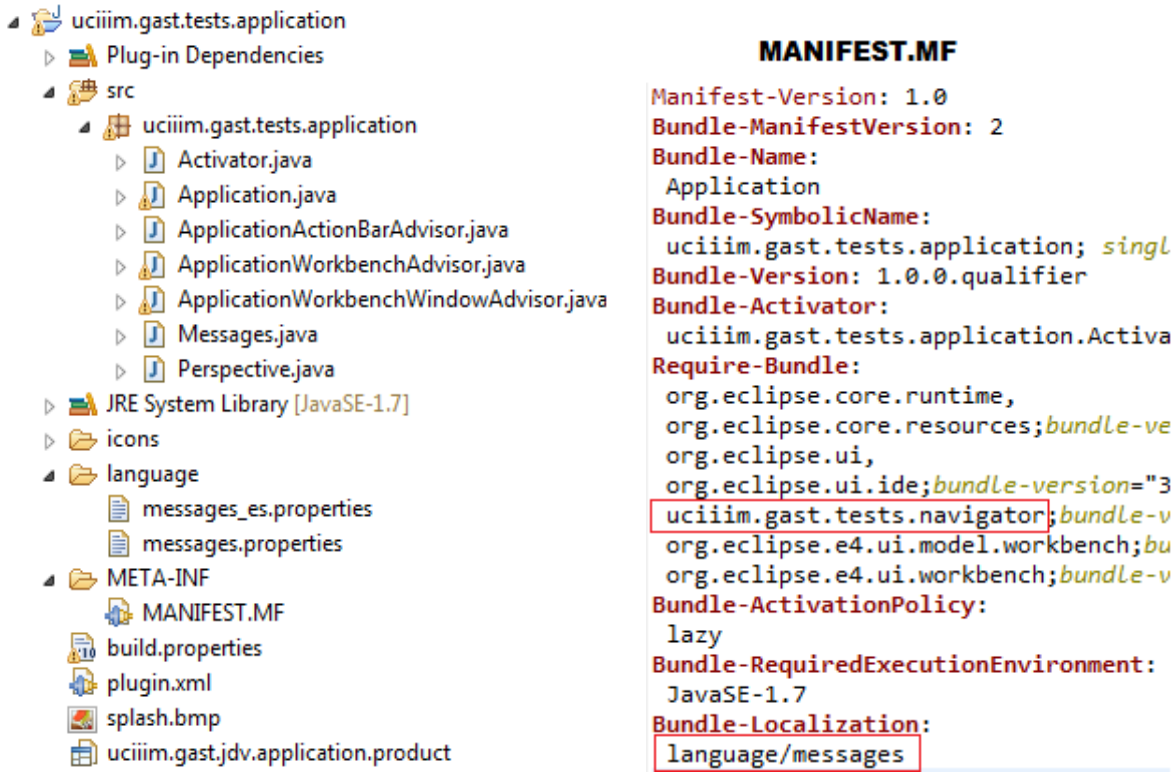


Figura 18 – Estructura y fichero MANIFEST.MF del plugin Aplicación

Se trata del plugin que define el producto de la aplicación, a partir del cual se genera el ejecutable de ésta. Contiene las clases que gestionan el ciclo de vida y además tiene como dependencia al plugin *Navegador*, con lo cual es quien crea la vista de *Proyectos*.

Su estructura está compuesta por:

- **src:** es la carpeta que almacena las clases Java. Se compone de un solo paquete y contiene un total de 7 clases:
    - **Activator:** inicializa el plugin.
    - **Application*.java:** estas clases definen el ciclo de vida de la aplicación, así como su configuración.
    - **Messages:** clase que almacena las variables usadas a la hora de internacionalizar la aplicación.
    - **Perspective:** define la perspectiva de la aplicación, con la vista de Proyectos y la de Esquema abiertas, y definiendo el espacio del para los editores.
- **icons:** iconos usados por esta aplicación, como son el icono del ejecutable o el de la ventada y diálogos del entorno.
- **language:** ficheros *.properties* que configuran los textos o mensajes usados en el plugin en diferentes idiomas.
- **MANIFEST.MF:** (Figura 18) fichero de configuración que define el nombre del plugin, qué otros plugins se tienen como dependencia, en este caso el del *Navegador*, y también en que ruta se encuentran los ficheros usados para la internacionalización del plugin.
- **build.properties:** fichero que define las carpetas y recursos que se exportan, en este caso a la hora de generar el producto.
- **plugin.xml:** fichero que configura las extensiones, en este caso el nombre de la aplicación, la definición del producto y la perspectiva por defecto.
- **splash.bmp:** se trata de la imagen que aparece en pantalla cuando se arranca la aplicación.
- **uciiim.gast.jdv.applicaction.product:** es el fichero que configura el producto a generar, que contendrá todas las dependencias, tanto las generadas en esta aplicación como las de Eclipse usadas en todas ellas. Define además el nombre del ejecutable y el icono a usar.

3.2.1.2 Navegador

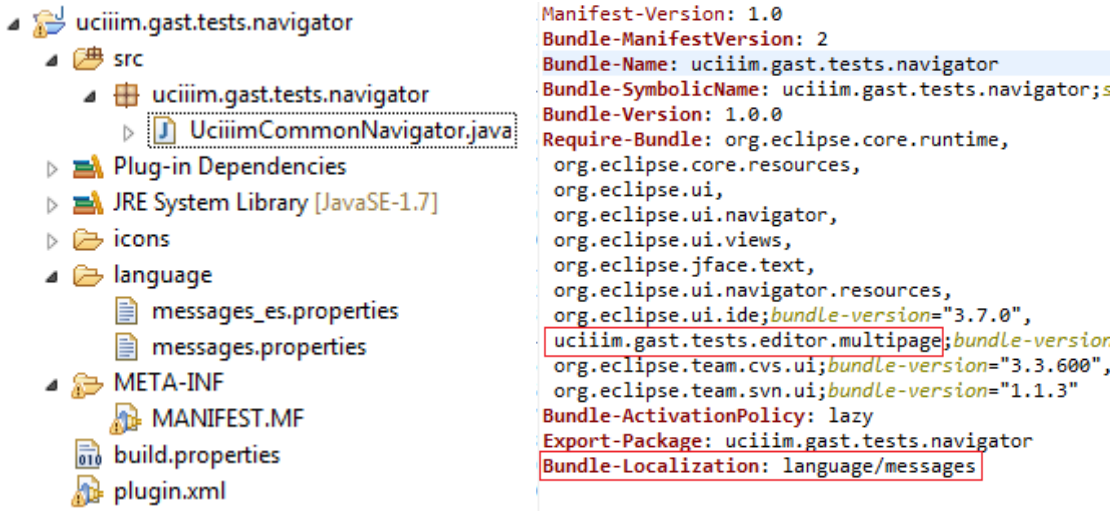


Figura 19 - Estructura y fichero MANIFEST.MF del plugin Navegador

Es el plugin que define la vista de *Proyectos*. Esta se trata de un navegador de proyectos que contiene un árbol SWT, en el cual se muestran la estructuran de proyectos que almacena el espacio de trabajo configurado, y que proporciona opciones a través de un menú superior en la propia vista y un menú contextual.

Este plugin, como se observa en la Figura 19, únicamente se compone de una clase Java, por lo que toda la funcionalidad radica en el fichero *plugin.xml*. Está compuesto por:

- **src:** es la carpeta que almacena la única clase Java del plugin:
    - **UciiimCommonNavigator:** extiende las funciones del navegador para capturar el evento de doble click sobre el árbol de proyectos y lanzar el editor multi-página XML/Gráfico si se trata de un fichero de extensión *.xml*.
- **icons:** iconos usados por esta aplicación, como los de las opciones crear nuevo proyecto o archivo de test y sus diálogos.
- **language:** ficheros *.properties* que configuran los textos o mensajes usados en el plugin en diferentes idiomas.
- **MANIFEST.MF:** (Figura 19) fichero de configuración que define el nombre del plugin, qué otros plugins se tienen como dependencia, en este caso el del *Editor Multi-página*, y también en que ruta se encuentran los ficheros usados para la internacionalización del plugin.

- **build.properties:** fichero que define las carpetas y recursos que se exportan, en este caso a la hora de generar el producto.

- **plugin.xml:** fichero que configura las extensiones, y en este caso el que concentra toda la información y configuración del plugin. Mediante puntos de extensión se configura la vista de *Proyectos*, con sus menús, opciones, proveedores que definen qué y cómo se muestran los elementos en el árbol de proyectos, etc.

3.2.1.3 Editor Multi-página

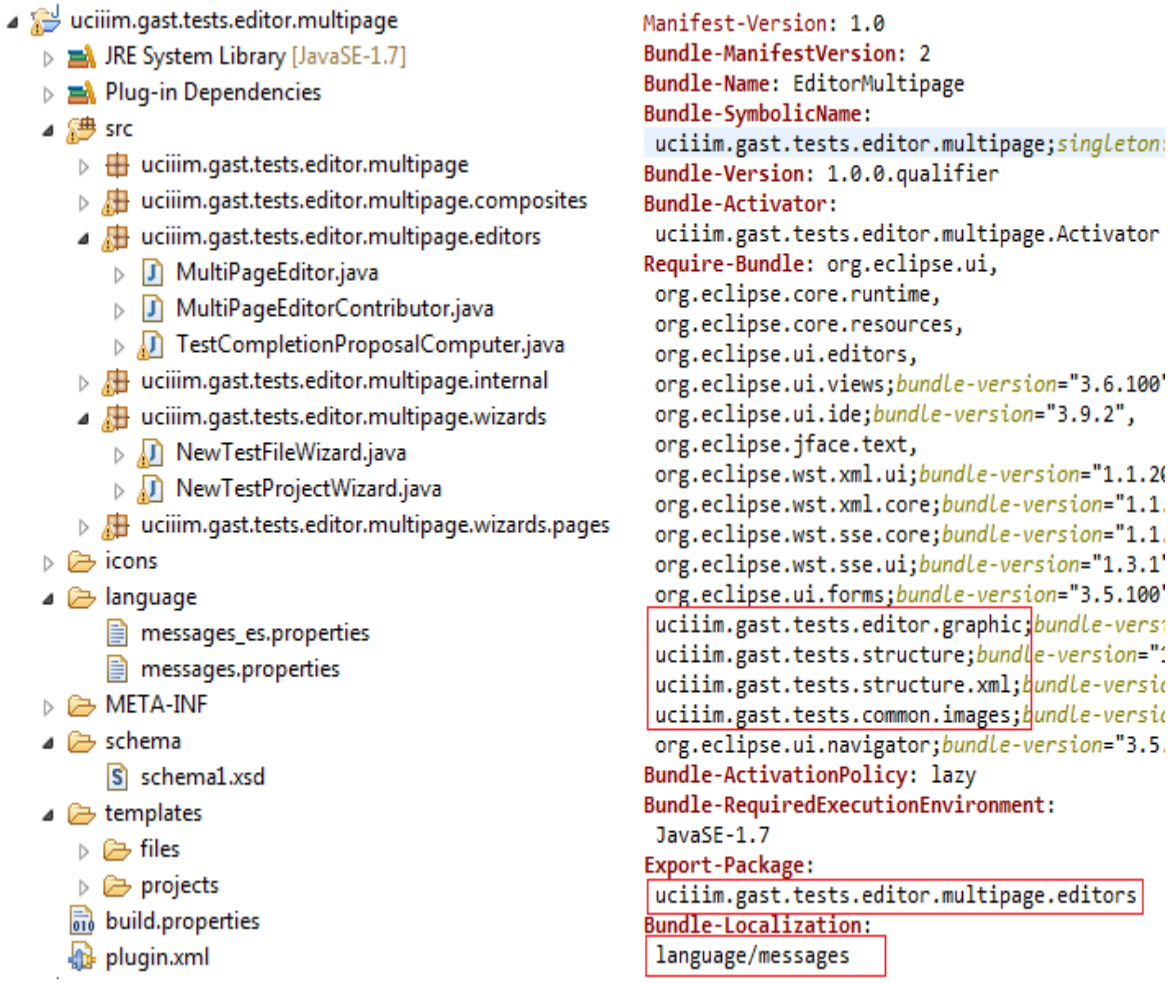


Figura 20 - Estructura y fichero MANIFEST.MF del plugin Editor Multi-página

Es el plugin que define el editor XML multi-página y se trata de uno de los más complejos de la aplicación, que incluye dependencias con hasta cuatro plugins de la aplicación.

Se compone de:

- **src:** es la carpeta que almacena los seis paquetes en los cuales se reparten las 16 clases Java del plugin, entre las cuales destacan:

    - **MultiPageEditor:** se trata del editor XML multi-página, al cual se le añade la página de diseño definida en el plugin *Editor Gráfico*. Será quien reciba el contenido del archivo XML a representar, cree la instancia de tipo *ITest* mediante el plugin *Estructura de Test* XML, y se la mande a la página de diseño; y por lo tanto quien reciba de ésta página la instancia de *ITest* modificada y la represente en las páginas de *Árbol* y *Fuente* XML.

    - **NewTestFileWizard:** se trata de la clase que configura las dos páginas del diálogo de creación de un nuevo archivo de test.

    - **NewTestProjectWizard:** se trata de la clase que configura las dos páginas del diálogo de creación de un nuevo proyecto de test.

- **icons:** iconos usados por el fichero *plugin.xml* de esta aplicación, los cuales no pueden ser incluidos en el plugin *Imágenes Comunes*, como sí lo son otros iconos usados en este plugin.

- **language:** ficheros *.properties* que configuran los textos o mensajes usados en el plugin en diferentes idiomas.

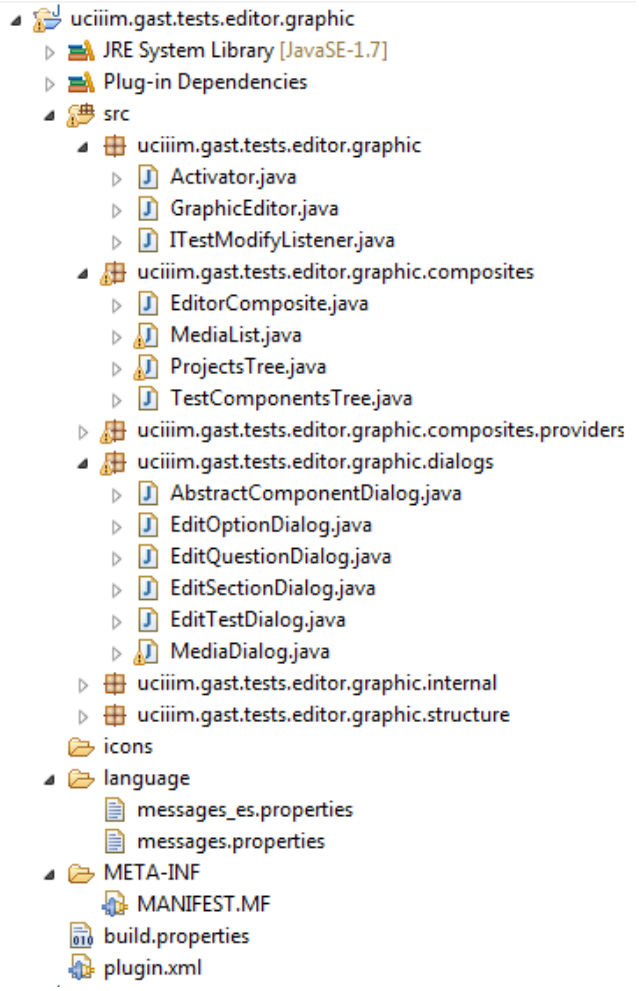
- **MANIFEST.MF:** (Figura 20Figura 19) fichero de configuración que define el nombre del plugin, qué otros plugins se tienen como dependencia, en este caso hasta un total de cuatro (*Editor Gráfico, Imágenes Comunes, Estructura de Test y Estructura de Test XML*), la ruta se encuentran los ficheros usados para la internacionalización del plugin, y en este caso qué paquete se exporta para ser usado por quienes incluyan como dependencia a este plugin, el caso del plugin *Navegador*.

- **schema1.xsd:** es el schema que define la estructura de los ficheros XML de test, el cual se configura en el fichero *plugin*.xml de este plugin para que el editor de fuente XML lo utilice.

- **build.properties:** fichero que define las carpetas y recursos que se exportan, en este caso a la hora de generar el producto.

- **plugin.xml:** fichero que configura las extensiones, tales como el editor, las opciones de los menús, los diálogos de creación de proyecto y archivo de test, el fichero XSD utilizado por el editor, etc.

3.2.1.4 Editor Gráfico




```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Graphic
Bundle-SymbolicName:
 uciiim.gast.tests.editor.graphic;singleton:
Bundle-Version: 1.0.0.qualifier
Bundle-Activator:
 uciiim.gast.tests.editor.graphic.Activator
Require-Bundle: org.eclipse.core.runtime,
 org.eclipse.ui;bundle-version="3.7.0",
 org.eclipse.ui.forms;bundle-version="3.5.10
 org.eclipse.jface.text;bundle-version="3.8.
 org.eclipse.core.resources;bundle-version="
 org.eclipse.ui.navigator;bundle-version="3.
 org.eclipse.ui.ide;bundle-version="3.9.2",
 uciiim.gast.tests.common.images;bundle-vers
 uciiim.gast.tests.structure;bundle-version=
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment:
 JavaSE-1.7
Export-Package:
 uciiim.gast.tests.editor.graphic
Bundle-Localization:
 language/messages
```

Figura 21 - Estructura y fichero MANIFEST.MF del plugin Editor Gráfico

Es el plugin que contiene el editor gráfico y por tanto el más complejo de todos. Guarda dos dependencias con otros dos plugins de la aplicación.

Está compuesto por:

- **src:** es la carpeta que almacena los seis paquetes en los cuales se reparten las 27 clases Java del plugin, entre las cuales destacan:
    - **GraphicEditor:** se trata de la página que contiene el editor gráfico. Se encarga de la configuración de ésta página y contiene una instancia del componente *EditorComposite*, descrito a continuación, al cual se le pasará la instancia *ITest* recibida por el editor multi-página, y del cual la recibirá ya modificada para devolvérsela a éste.

  - **EditorComposite:** es el editor gráfico en sí. Su estructura se construye a partir del elemento principal definido por el plugin Eclipse Forms, el cual se usa para crear las secciones y demás componentes gráficos. Incluye los menús que este editor tiene, así como las llamas a los diálogos que permiten modificar la instancia *ITest* almacenada.

  - **MediaList, ProjectsTree y TestComponentsTree:** son quienes crean las tablas y árboles que aparecen en los diálogos que se definen a continuación. Muestran la información estructurada definida en el elemento *Itest*, que no es más que la representación del contenido del archivo XML.

  - **AbstractComponentDialog:** como su propio nombre indica, se trata de una clase abstracta que define la estructura de un diálogo. En concreto, las clases que la extienden y que se hayan en su mismo paquete, denominadas **Edit_*_Dialog**, son las que permiten editar de gráficamente, de forma fácil y sencilla, el contenido de la estructura *Itest*, y por lo tanto la del propio fichero XML.

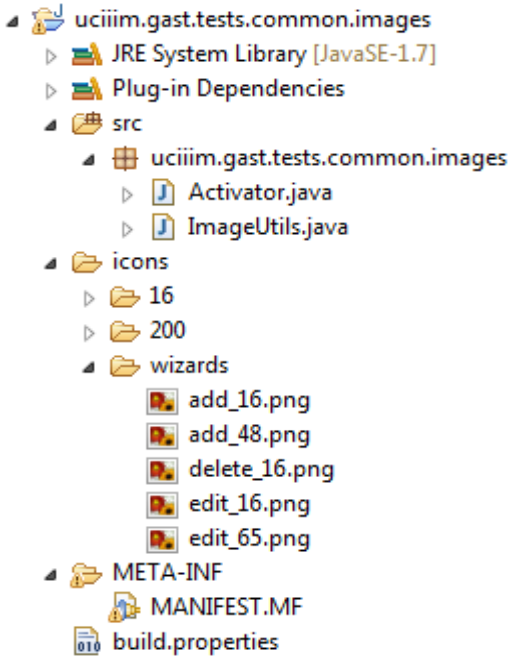
- **language:** ficheros *.properties* que configuran los textos o mensajes usados en el plugin en diferentes idiomas.

- **MANIFEST.MF:** Figura 19(Figura 21) fichero de configuración que define el nombre del plugin, qué otros plugins se tienen como dependencia, en este caso dos: *Imágenes Comunes y Estructura de Test*), la ruta se encuentran los ficheros usados para la internacionalización del plugin, y qué paquete se exporta para ser usado por quienes incluyan como dependencia a este plugin, el caso del plugin *Editor Multi-página*.

- **build.properties:** fichero que define las carpetas y recursos que se exportan, en este caso a la hora de generar el producto.

- **plugin.xml:** fichero que configura las extensiones.

3.2.1.5 Imágenes comunes




```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Images
Bundle-SymbolicName:
 uciiim.gast.tests.common.images
Bundle-Version: 1.0.0.qualifier
Bundle-Activator:
 uciiim.gast.tests.common.images.Activator
Require-Bundle: org.eclipse.core.runtime,
 org.eclipse.ui.ide;bundle-version="3.9.2",
 org.eclipse.swt;bundle-version="3.102.1",
 org.eclipse.jface;bundle-version="3.9.1",
 org.eclipse.ui.workbench;bundle-version="3
Bundle-ActivationPolicy: lazy
Export-Package:
 uciiim.gast.tests.common.images
```

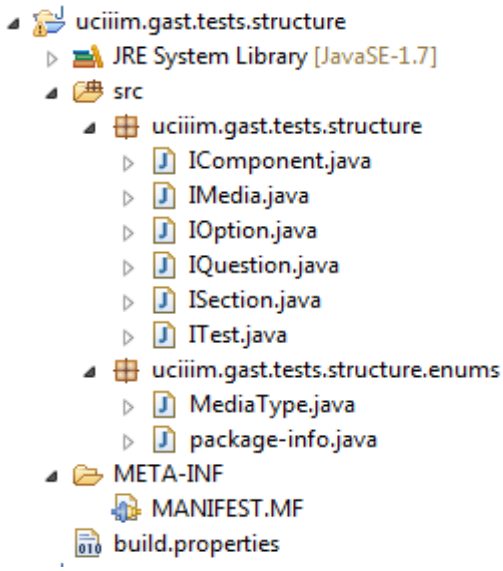
Figura 22 - Estructura y fichero MANIFEST.MF del plugin Imágenes Comunes

Plugin que no tiene dependencia con ningún otro plugin de la aplicación, ya que simplemente almacena las imágenes comunes usadas en los plugins *Editor Multi-página* y *Editor Gráfico*.

Su estructura es:

- **src:** es la carpeta que almacena las dos clases Java que forman el plugin. Destacar:
    - **ImageUtils:** se trata de una clase de utilidades, como su propio nombre indica, la cual contiene métodos estáticos que permiten obtener los iconos que el plugin almacena y en realizar modificaciones a estos, como cambiarles el tamaño.
- **icons:** carpeta que almacena los iconos e imágenes, de distintos tamaños, que pueden ser referenciados desde otros plugins.
- **MANIFEST.MF:** (Figura 22) fichero de configuración que define el nombre del plugin, y el nombre del paquete que se exporta para ser usado por los demás plugins.
- **build.properties:** fichero que define las carpetas y recursos que se exportan, en este caso a la hora de generar el producto. Importante en este caso será la carpeta *icons* y sus sub-carpetas e iconos.

3.2.1.6 Estructura de test




```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Tests Structure
Bundle-SymbolicName:
 uciiim.gast.tests.structure
Bundle-Version: 1.0.0.qualifier
Bundle-RequiredExecutionEnvironment:
 JavaSE-1.7
Export-Package:
 uciiim.gast.tests.structure,
 uciiim.gast.tests.structure.enums
```

Figura 23 - Estructura y fichero MANIFEST.MF del plugin Estructura de Test

Define la estructura diseñada por esta aplicación para almacenar el contenido de test, y a simple vista se puede observar la sencillez que aporta, algo que cumple con los requisitos definidos por esta aplicación.

Este plugin es independiente de los demás plugins, incluidos los plugins de Eclipse, ya que no se trata de un plugin de Eclipse RCP como tal.

En cambio, los plugins *Estructura de Test XML, Editor Multi-página y Editor Gráfico*, si van a tener a este plugin como dependencia.

Su estructura es:

- **src:** es la carpeta en la cual se definen los dos paquetes que contienen las 7 clases Java que forman el plugin. La definición de estas y su relación entre sí se puede ver en el apartado 3.1.1.1, en el cual se detalla el diseño y la estructura de estas clases.

- **MANIFEST.MF:** Figura 19(Figura 23) fichero de configuración que define el nombre del plugin, además de enumerar el nombre de los paquetes que se exportan para ser usados por los demás plugins.

- **build.properties:** fichero que define las carpetas y recursos que se exportan, en este caso a la hora de generar el producto.

3.2.1.7 Estructura de test XML

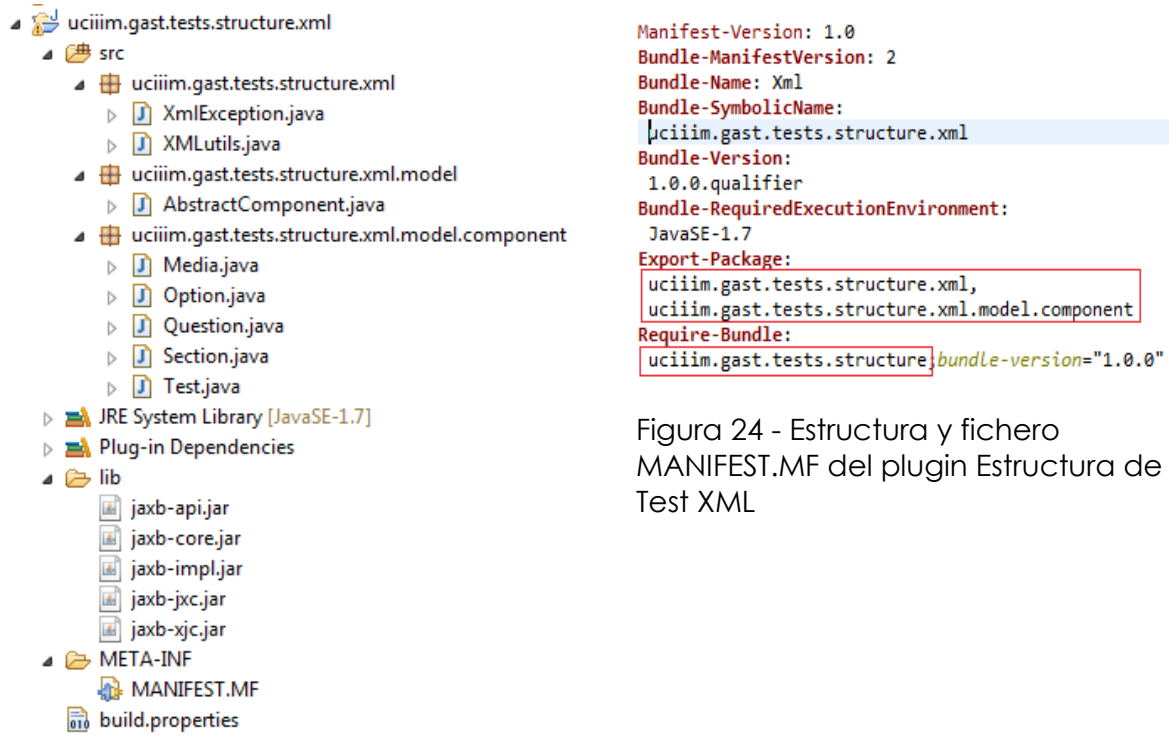


Figura 24 - Estructura y fichero MANIFEST.MF del plugin Estructura de Test XML

Es el plugin que implementa la funcionalidad definida en el plugin *Estructura de Test*, al cual tendrá como dependencia. Sus clases definen la estructura de los test como si fuesen elementos XML, ya que se los nombres y atributos de las clases se configuran para que definan cada uno de ellos a un elemento XML.

En este caso tampoco se trata de un plugin RCP, ya que no guarda dependencia con ningunos de los plugins definidos por Eclipse. De ahí que pueda ser reusado en otros entornos Java.

Su estructura es:

- **src:** es la carpeta en la cual se definen los tres paquetes que contienen las 8 clases Java que forman el plugin. La definición de estas y su relación entre sí se puede ver en el apartado 3.1.2, en el cual se detalla el diseño y la estructura de estas clases. Además, existen otras dos clases dentro de este plugin que proporcionan otro tipo de opciones:
    - **XmlException**: define una excepción a lanzar cuando no se pueda importar o exportar el contenido de un fichero XML dado. Si el fichero no está bien formado, los métodos que se encargan de analizar su contenido y convertirlo a Java, o viceversa, lanzarán una excepción de este tipo.

- **XMLutils**: define los métodos estáticos que permiten analizar un fichero de test en formato XML y convertirlo a un objeto *Test* de java, o bien lo contrario, que a partir de un objeto de tipo *Test* se pueda generar un fichero XML que plasme el contenido Java en este formato. Estas operaciones se realizan mediante las opciones que proporciona JAXB.

- **lib:** directorio que contiene las librerías de JAXB usadas en la clase *XMLutils* para analizar el contenido de un fichero XML y convertirlo a una instancia de la interfaz Java denominada *Itest*, y viceversa.

- **MANIFEST.MF:** Figura 19(Figura 24) fichero de configuración que define el nombre del plugin, además de enumerar el nombre de los paquetes que se exportan para ser usados por los demás plugins y las dependencias con otros plugins.

- **build.properties:** fichero que define las carpetas y recursos que se exportan. No olvidar incluir la carpeta *lib* con todas las librerías de JAXB.

3.2.2 Comunicación XML-Java

A continuación se define el orden lógico de comunicación e interacción entre cada una de las partes del proyecto conseguir que un fichero almacenado en el espacio de trabajo en formato XML llegue a ser representado en la interfaz gráfica Java, éste pueda ser modificado en ella y posteriormente volcado de nuevo en el fichero XML.

Estos pasos se resumen en:

1. El fichero se encuentra almacenado en el espacio de trabajo bajo un proyecto, y por lo tanto esta visible en la vista de Proyectos, incluida en el plugin Navegador.

2. Este fichero será abierto usando el editor configurado para su extensión, en este caso XML, y por lo tanto será representado en la página de código fuente XML que forma parte de editor, y que se encuentra en el plugin Editor Multi-página.

3. Ese contenido, en texto plano, será analizado utilizando el plugin de Estructura de Test XML, quien mediante el uso de JAXB creará un objeto Java de tipo ITest, el cual se trata de una de las interfaces definidas en el plugin Estructura de Test.

4. Una vez se dispone de un objeto Java en el editor multi-página, éste se lo hará llegar a su página de diseño, en la cual se representará el fichero de test en forma de pre-visualización.

5. Este editor gráfico permite modificar la estructura de ese objeto ITest, y esos cambios efectuados en el editor gráfico se volcarán de nuevo mediante el objeto ITest a la página de código fuente XML de editor multi-página.

6. El editor volverá a hacer uso del plugin que define una estructura de ITest en formato XML, y de este modo convertirá de nuevo el objeto en código XML.

7. Ese código se volverá a volcar contra el fichero de texto.

De esta forma será como se comuniquen los diferentes plugins, y de ahí se obtienen las dependencias necesarias entre ellos de las cuales se ha hablado con anterioridad.

A continuación se muestra gráficamente el proceso descrito anteriormente.

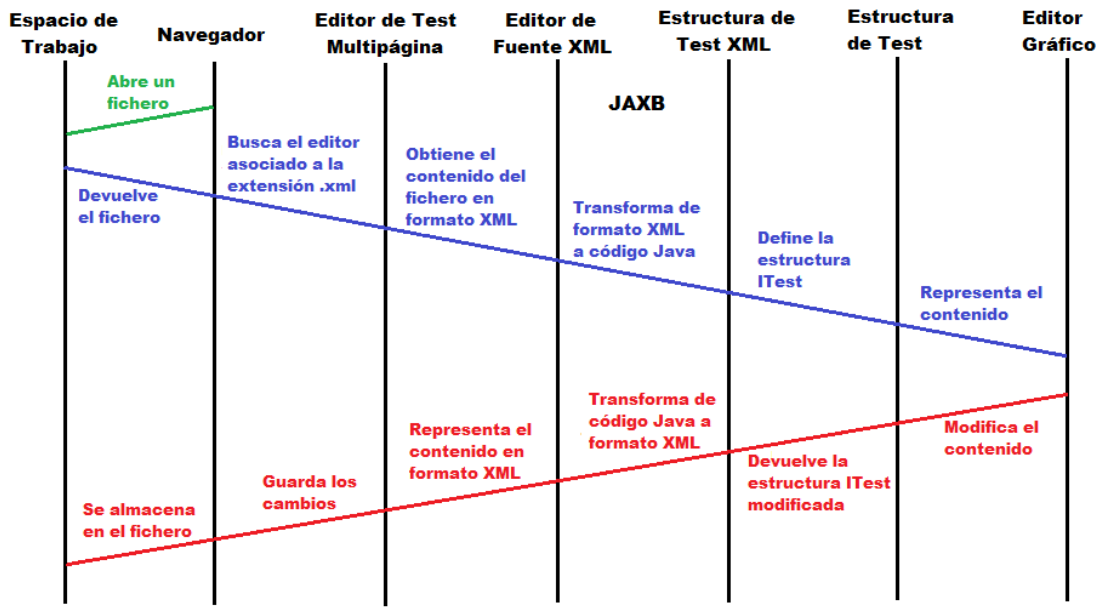


Figura 25 – Proceso de comunicación (XML-Java)

3.2.3 Modificación de la aplicación

Uno de los objetivos del desarrollo de la aplicación era el crear un sistema fácilmente modificable, de forma que pudiese ser actualizado de manera sencilla. Por eso aquí se enumeran una serie de modificaciones que podrían llevarse a cabo y se detalla cómo deberían realizarse estos cambios, que clases se deberían tocar y cómo afectaría cada uno de los cambios.

Puesto que la aplicación se encuentra modularizada en partes independientes, se podrá de manera fácil y sencilla modificar la funcionalidad de esta.

En algunos casos se trataría de modificar las dependencias existentes entre los plugins, para eliminar las actuales e introducir las nuevas. En otros casos podría tratarse de añadir o quitar funcionalidad a un determinado plugin, como pueden ser opciones de menú, cambiar imágenes o modificar textos, añadir nuevas vistas o definir nuevas perspectivas, etc., y esta se haría modificando el fichero *plugin.xml* de los plugins. Y en otros casos de podría directamente modificar el código Java para cambiar determinados comportamientos, mejorar una implementación o cualquier otra modificación que se quiera implementar.

A continuación se enumeran una serie de puntos a tener en cuenta a la hora de modificar el funcionamiento de esta aplicación.

3.2.3.1 Cambio de la estructura definida de Test

Si se desea modificar el plugin que define la **Estructura de Test** con la finalidad por ejemplo de incluir nuevos tipos de respuesta, nuevos atributos, o simplemente modificar su estructura**,** bastaría con cambiar las interfaces que éste define.

Ahora bien, este cambio se traduciría en modificaciones en todos los plugins que dependan de este, como son:

- **Estructura de Test XML:** implementar su equivalente en Java de manera que JAXB pueda seguir haciendo la conversión XML-Java de manera eficiente y sin errores.

- **Editor Multi-página:** en este caso solo si la interfaz base modifica su nombre (*ITest*).

- **Editor Gráfico:** debería modificar su funcionamiento en el orden de su correspondencia gráfica, por tanto dependería del tamaño de las modificaciones efectuadas en la estructura y de cómo de complejas fuesen estas. Al final sería el que mayor carga de trabajo generaría, ya que habría que crear nuevos widgets, nuevas opciones, nuevos diálogos, etc.

3.2.3.2 Cambio de formato de los ficheros de test

Si se decide dejar de usar XML como formato a la hora de almacenar el contenido de los ficheros de test, para cambiarlo por ejemplo por un formato de texto plano basado en elementos de atributo/valor, la modificación se centraría en dos plugins:

- **Estructura de Test XML:** este dejaría de ser utilizado como tal, por lo que debería modificarse o generar uno nuevo que soportase el nuevo formato de los ficheros de test. Tendría que seguir implementando al plugin **Estructura de Test**, pero su estructura Java no usaría JAXB ni se correspondería con elementos XML.

- **Editor Multi-página:** el editor de código fuente XML se dejará de usar, para pasar a definir el nuevo formato de los test. Por su parte, se debería analizar si la página de árbol continuaría teniendo sentido, ya que esta suele acompaña al formato XML. En cuanto al editor gráfico, ya es sabido que podría funcionar en solitario, siempre y cuando le llegue una instancia de tipo *ITest* como entrada.

3.2.3.3 Cambio del editor gráfico

Si se decide modificar el plugin del **Editor Gráfico,** para por ejemplo eliminar las dependencias con Eclipse Forms y re-implementar la pre-visualización de los test mediante el uso únicamente de SWT y JFace, o simplemente implementar un nuevo editor gráfico, éste debe recibir como entrada una instancia de *ITest* y generar como salida un objeto de que implemente esa misma interfaz.

De esta forma se asegura que el resto del sistema seguirá funcionan correctamente, sin necesidad de implementar modificación alguna.

3.2.3.4 Actualizar el Navegador

Si se desean añadir nuevas funcionalidades a la vista de Proyectos, ya sea al árbol de contenido de los recursos (proyectos, carpetas, ficheros…), al menú superior de la vista o al menú contextual de esta, se deberá hacer editando el fichero *plugin.xml*, ya que como se ha citado con anterioridad cuando se ha definido este plugin, su funcionalidad está centralizada en este fichero.

En él se define que recursos se muestran y como, que extensiones de otros plugins se añaden para aportar nuevas funcionalidades, que opciones aparecen tanto en sus opciones del menú superior como en el menú contextual.

El manejo del fichero *plugin.xml* y la forma en cómo se añaden funcionalidades en Eclipse suele ser sencilla, pero en este proyecto en concreto concretamente resulta muy complicada de entender, incluso para usuarios avanzados a la hora de crear plugins y aplicaciones RCP.

Modificar el navegador podría ser útil por ejemplo a la hora de definir una estructura obligatoria para los proyectos de test. Se deberían modificar los proveedores del árbol de proyectos, para que solo aceptasen esa estructura, y además cambiar opciones de menú y realizar comprobaciones que aseguren dicha estructura.

3.2.3.5 Modificar las opciones en el menú y la toolbar

Al igual que en caso anterior del Navegador, las opciones del menú superior y la toolbar de la aplicación se definen en el fichero *plugin.xml*, pero en este caso en el plugin del **Editor Multi-página**.

El fichero es mucho más sencillo y fácil de comprender que el anterior, por lo que añadir, modificar o borrar opciones de no debe ser complicado. Basta con abrir el fichero, buscar los puntos de extensión *"org.eclipse.ui.commands"* y *"org.eclipse.ui.menus",* y ver como primero se definen los comandos y después se usar para crear los diferentes menús.

Las opciones pueden ser propias de Eclipse, mediante el identificador del comando que define la opción, que puede estar siendo añadida gracias a una nueva dependencia con un plugin concreto.

3.2.3.6 Cambio de los iconos de la aplicación

A pesar de disponer de un plugin común de imágenes y clases de utilidades para manejar estas, únicamente los plugins que forman el editor (**Editor Multi-página** y **Editor Gráfico**) van a tenerlo como dependencia, y por lo tanto a usar sus imágenes.

Esto se debe a que los otros dos plugins que añaden iconos, **Aplicación** y **Navegador**, necesitan disponen de los iconos en su misma ubicación para funcionar correctamente.

Por tanto, si en el editor requiere cambiar los iconos que muestra, bastará con modificar el plugin de **Imágenes Comunes.**

3.2.3.7 Configuración de nuevos idiomas

En los requisitos funcionales se hablaba de la posibilidad de disponer de diferentes versiones de la aplicación, en función por ejemplo del idioma. Se ha detallado anteriormente que la implementación de la aplicación se ha configurado de manera que por defecto se podría arrancar en dos idiomas: español e inglés.

Además, gracias a la filosofía y las herramientas que Eclipse RCP proporciona, podrían incluirse y configurarse otro tipo de idiomas. Esta funcionalidad está disponible para que cualquier usuario pueda extenderla e implementar una extensión que añada un nuevo idioma.

En el anexo 8.4.1 se explica cómo se han definido las clases necesarias y los cambios efectuados en los ficheros de configuración para que esto funcione correctamente. A continuación bastaría con añadir un nuevo fichero

messages_[language].properties con los textos a sobrescribir de la versión por defecto, en este caso el inglés.

3.2.4 Reutilización de los plugins

Uno de los objetivos marcados a la hora empezar a desarrollar de este proyecto era que el código generado pudiese ser fácilmente reutilizable. Esta premisa no se basa únicamente en la reutilización a nivel de código Java (algo que también se ha conseguido tal y como se ha detallado con anterioridad), sino que se decía que cada uno de los módulos generados debía tener sentido por sí mismo y exportar funcionalidades que pudiesen ser reutilizadas por otras aplicación y en entornos diferentes.

Relativo a ello se ha conseguido que todos los plugins de esta aplicación puedan ser reusados en otros desarrollos, a excepción del que define la aplicación en sí (plugin Aplicación), tal y como se detalla a continuación:

- **Estructura de Test:** este plugin no se trataba en sí de un complemento exclusivo de la tecnología RCP, sino que por su definición puede ser usado en cualquier implementación Java a modo de librería. No contiene dependencias con ningún otro plugin, lo que le hace ser independiente de los demás.

- **Estructura de Test XML:** este caso es similar al anterior, con la única peculiaridad que sí que depende de este último, pero es la única de sus dependencias con otros plugins. De este modo podría usarse también en desarrollos Java diferentes a RCP, claramente para analizar los ficheros de test con el formado definido en esta aplicación.

- **Imágenes comunes:** al tratarse únicamente de iconos y clases de utilidades, y no tener dependencias con ningún otro plugin de la aplicación, podría reusarse incluso de manera directa, sin necesidad de ser modificado.

- **Navegador:** en este caso sí sería necesario modificar la configuración y la implementación del plugin, aunque el cambio se reduce a suprimir la dependencia con el plugin del Editor Multi-página y eliminar el código que le referencie, que se reduce a la clase Java que define el Navegador, dentro de la cual se escucha el evento del doble-click de ratón en la vista de Proyectos y en caso de que se haya realizado sobre un fichero con extensión XML, el archivo se visualizará usando el editor multi-página. De este modo, el plugin Navegador podría ser reusado en cualquier otra aplicación Eclipse RCP.

- **Editor Multi-página:** su fundamento es la representación de ficheros XML en las páginas de código fuente y árbol, algo que ya proporciona el plugin de Eclipse que define un editor XML con esas páginas. Sin embargo, en esta aplicación se ha extendido para incluir la página de diseño y pre-visualización, por lo que cualquier uso de este plugin sin el plugin Editor Gráfico no tendría sentido alguno. Por tanto, su reutilización tal y como está implementado se limita a ir acompañado siempre del editor gráfico, además de ambos plugins de estructura y el de imágenes comunes, con los que guarda dependencia.

- **Editor gráfico:** este plugin guarda dependencia con el plugin de imágenes, algo que podría sustituirse por otra implementación distinta, Sin embargo, el que no puede ser eliminado de su lista de dependencias es el plugin que define las interfaces de la Estructura de Test, ya que su diseño a la hora de crear, modificar y representar los cuestionarios está exclusivamente diseñado para que reciba como entrada y genere como salida una instancia de tipo *ITest*. De este modo, podría reutilizarse en cualquier otro desarrollo RCP que soportase el formato definido por el plugin de Estructura de Test. Incluso, con una pequeña modificación, podría llegar a ser utilizado sin necesidad de tener que tratarse de una página de un editor, ya que la representación gráfica de la información se concentra en una clase definida como un componente gráfico de *SWT*. Esta clase se ha definido en el apartado 3.2.1.4 de este mismo capítulo y tiene por nombre *uciiim.gast.tests.editor.graphic.composites.EditorComposite*.

3.3 Pruebas

Se han llevado una serie de pruebas manuales para verificar el correcto comportamiento de la aplicación.

3.3.1 Pruebas manuales

Estas pruebas se han llevado a cabo con el fin de verificar todas las funcionalidades que ofrece la aplicación. Para ello, se han llevado a cabo una serie de pruebas manuales para verificar que todas y cada una de las implementaciones hechas funcione correctamente.

La batería de pruebas se compone de un total de 103 pruebas individuales, separadas en grupos o secciones. En el anexo 8.2.1 se puede visualizar la lista completa de pruebas y sus resultados, y a continuación se muestra una tabla con el resumen de las mismas.

Grupo	Resumen de las pruebas	Total	OK
Ciclo de Vida	• comportamiento de la aplicación de inicio a fin • configuración del espacio de trabajo • opciones de reinicio.	11	Todas
Navegador de Proyectos	• visualización del árbol de recursos (proyectos, carpetas, archivos…) en la vista de Proyectos • opciones del menú contextual • opciones de creación de recursos (proyectos y ficheros de test) • configuración de repositorios de contenido • opciones del menú superior de la vista	28	Todas
Editor XML/Gráfico Multi-página	• visualización del fichero XML en el editor multi-página • representación, opciones y diálogos del editor gráfico • representación y opciones del editor en árbol • representación y opciones del editor en formato XML fuente • representación e interacción entre ellos	36	Todas
Opciones de menú y toolbar	• opciones y diálogos del menú *Archivo* • opciones y diálogos del menú *Edición* • opciones y diálogos del menú *Buscar* • opciones y diálogos del menú *Ventana* • opciones, ventanas y vistas del menú *Ayuda* • opciones de la *toolbar*	28	Todas

Tabla 9 – Resumen de las pruebas manuales

Esta batería de pruebas ha permitido detectar errores relacionados especialmente con los diálogos de creación y edición de elementos del editor gráfico, y cómo modificaban estas las instancias del objeto *Itest*. Además, otro tipo de errores detectados en algunas de las opciones del menú superior y el menú contextual del navegador de proyectos. Una vez corregidos estos errores se ha vuelto a pasar la batería de pruebas hasta obtener la salida deseada.

Capítulo 4

Descripción de la aplicación

Este apartado resume el funcionamiento de la aplicación desarrollada. Se muestra en diferentes apartados cuáles son las características principales a tener en cuenta durante el uso de la misma, enumerando una serie de pasos a dar para probar sus principales opciones y hacerse una idea de su funcionamiento.

Sin embargo, se debe añadir que este apartado no se trata de un manual de usuario del entorno, sino más bien un resumen de este. En el anexo 8.1 se puede visualizar el documento que detalla todas las posibilidades y opciones que ofrece la aplicación, a modo de manual de usuario, que además se encuentra disponible para ser consultado desde una de las opciones de ayuda de la propia aplicación.

4.1 Inicio, configuración y ciclo de vida de la aplicación

Como se ha descrito en capítulos anteriores, el proyecto desarrollado se trata de una aplicación de escritorio, y como tal, cuenta con un archivo ejecutable. Se acompaña de las carpetas de plugins y configuración, además de otros archivos de inicialización, que componen la aplicación.

Los pasos para iniciar la aplicación, configurarla y controlar su ciclo de vida podrían resumirse en los siguientes:

- **Paso 1:** Iniciar la aplicación, mediante el fichero ejecutable *xmlGraphicTestEditor.exe* que se encuentra en la carpeta de la aplicación.

- **Paso 2:** Se abrirá una imagen en mitad de la pantalla, y a continuación, si es la primera vez que se arranca la aplicación, aparecerá el diálogo de configuración del espacio de trabajo, que no es más que una carpeta en la cual se crearán todos los proyectos de test. Se puede elegir un espacio de trabajo creado previamente, o definir uno nuevo. Además, permite elegir si este diálogo se ha de mostrar o no en futuros inicios de la aplicación.

- **Paso 3:** Tras elegir el espacio de trabajo, arrancará la aplicación con el aspecto por defecto, o con el aspecto anterior definido por el espacio de trabajo elegido, si no es la primera vez que se usa. Posteriormente se tratan puntos acerca de la aplicación gráfica.

- **Paso 4:** Una vez iniciada la aplicación, configurado el espacio de trabajo y vista su apariencia, continúa la prueba de su ciclo de vida. Para ello, se dan diferentes opciones:

    a. Reiniciar la aplicación, mediante la opción '*Archivo > Reiniciar*'. Se reiniciará la aplicación gráfica, con las opciones de espacio de trabajo elegidas en el paso 2.

    b. Cambiar el espacio de trabajo, mediante la opción '*Archivo > Conmutar espacio de trabajo*'. Se muestra el diálogo de selección de espacio de trabajo y si se acepta, se reinicia la aplicación también.

    c. Cerrar la aplicación, mediante la opción '*Archivo > Salir*''. La aplicación procederá a cerrarse. Si hay algún archivo en edición que no ha sido salvado, preguntará antes de cerrarse. También se puede cerrar mediante el aspa roja de la esquina superior derecha, como cualquier aplicación.

A continuación se muestra un diagrama de flujo con el ciclo de vida de la aplicación, y los diferentes recursos gráficos que se irán mostrando, de los cuales hemos hablado con anterioridad.

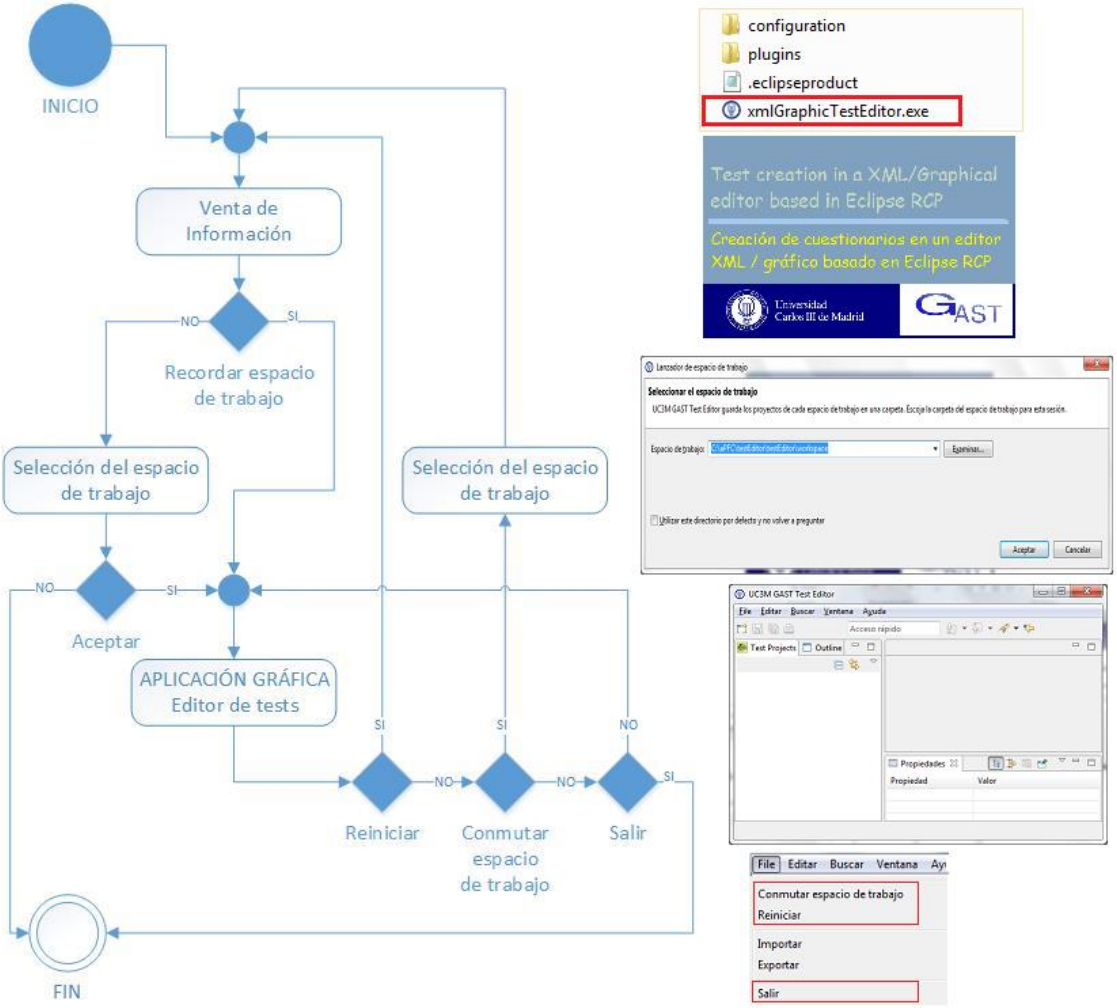


Figura 26 – Diagrama de flujo del ciclo de vida de la aplicación

4.2 Descripción del entorno

Una vez arrancada la aplicación se podrá observar la apariencia de esta. Para aquellos usuarios acostumbrados a utilizar Eclipse les resultará familiar, ya que la distribución de las vistas, menús y opciones son exactamente iguales a las del entorno de desarrollo. Estos no van a necesitan demasiadas explicaciones acerca del funcionamiento y desde su primer uso podrán crear proyectos, ficheros, utilizar el editor gráfico y sacarle partido a todas y cada una de las opciones disponibles.

Para aquellos usuarios no familiarizados con el mundo Eclipse o que nunca hayan usado un entorno similar, se ha comentado anteriormente que en el menú de ayuda de la aplicación existe la posibilidad de abrir una vista que muestra el manual de la aplicación.

En este apartado se intenta dar un enfoque global de cómo está diseñado el entorno, de que partes principales se compone, los menús, las vistas, etc…

La siguiente figura muestra una captura de qué encontrará un usuario en la aplicación la primera vez que abre ésta.

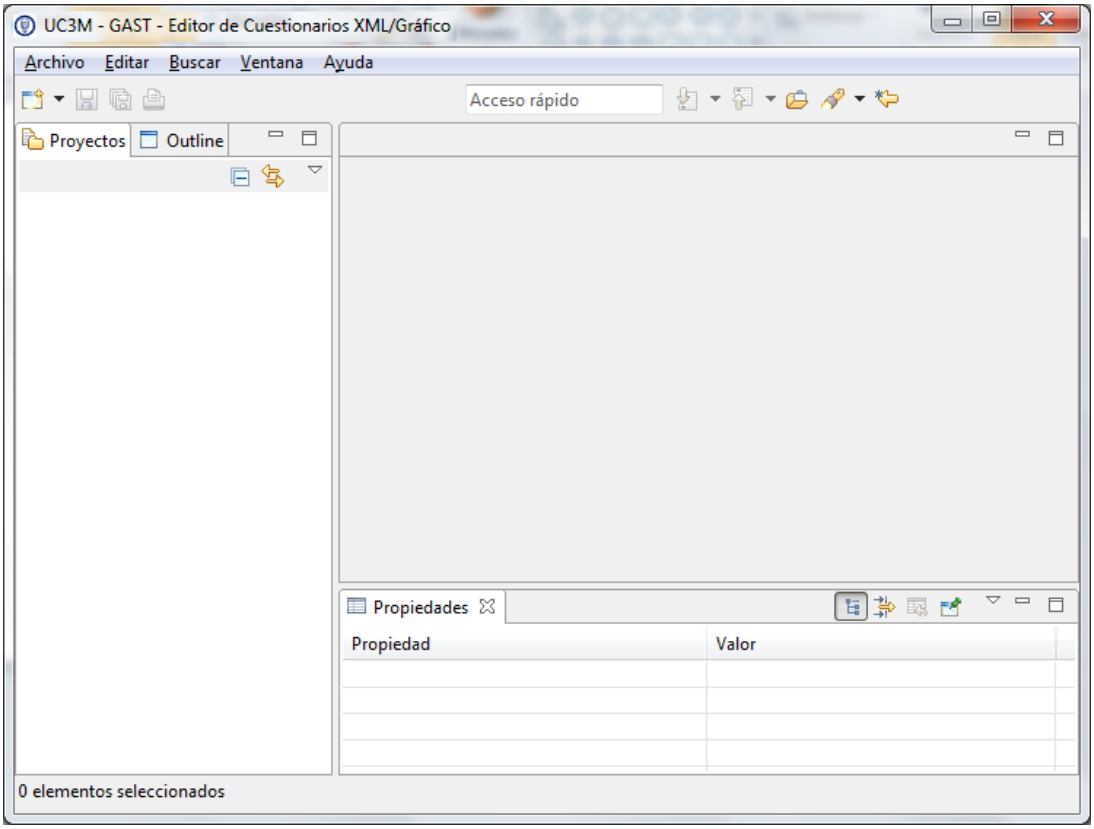


Figura 27 – Vista de la aplicación inicial

Como se indica, puede observarse cómo la aplicación es un entorno Eclipse con menos opciones de las que suele disponer éste. Se ha adaptado a las necesidades de este proyecto, pero sin perder su esencia.

Las partes de las que se compone son:

- **Navegador:** se trata de la vista denominada 'Proyectos', dentro de la cual se crean y se muestran los proyectos de test con todo su contenido: carpetas, ficheros de test, recursos gráficos (imágenes, audios y videos). Dispone de un menú contextual con numerosas opciones, que entre otras acciones permite crear, copiar, modificar o eliminar proyectos o parte de su contenido, exportar e importar estos, agruparlos en diferentes subcarpetas, gestionarlos mediante un repositorio compartido (control de versiones) CVS o SVN, ver sus propiedades… Todo ello se muestra en el manual de la aplicación.

Además, en su misma ubicación, por defecto estará la vista 'Outline', que muestra la estructura delos ficheros XML que se estén editando.

- **Menús:** se trata de la barra de opciones posicionada en la parte alta de la aplicación. Se distribuyen según su utilidad y agrupa opciones con un fin más o menos similar. Estos son:
    - **Archivo:** opciones de creación y edición de proyectos, carpetas, ficheros, etc. y opciones del ciclo de vida de la aplicación.
    - **Editar:** opciones de edición en los editores de texto.
    - **Buscar:** opciones de búsqueda dentro de la aplicación, ya sea ficheros o contenido de estos.
    - **Ventana:** opciones para cambiar la apariencia de la aplicación. Permite ocultar barras de herramientas, seleccionar que vistas se muestran y cuáles no, o también cambiar o reiniciar la perspectiva de la aplicación.
    - **Ayuda:** opciones para desplegar la ayuda de la aplicación, que contiene el manual de esta. También permite conocer detalles acerca de la versión y plugins que componen el entorno.
- **Toolbar:** se trata de la barra de herramientas inferior a los menús. En ella se muestran opciones acerca de la creación de proyectos, carpetas y ficheros, así como opciones de edición y guardado del contenido que se muestra en los editores. Son opciones disponibles en los menús pero debido a su alto nivel uso, se colocan aquí para hacerlas más intuitivas y rápidas.
- **Zona de editores:** se trata del espacio reservado a mostrar los editores, en este caso, el editor multi-página de test XML, pero también otros tipos de editores posibles. Suele estar dividida en dos de manera horizontal, y en su arte inferior suele disponer de otras vistas. En este caso, por defecto se muestra la vista de 'Propiedades'.
- **Barra de estado:** se trata de la barra de información que se encuentra en la parte inferior de la aplicación. En este caso mostrará información acerca de que está seleccionado en un determinado momento dentro de la aplicación.

A continuación se vuelve a mostrar la apariencia de la aplicación, pero con estas zonas diferenciadas en colores.

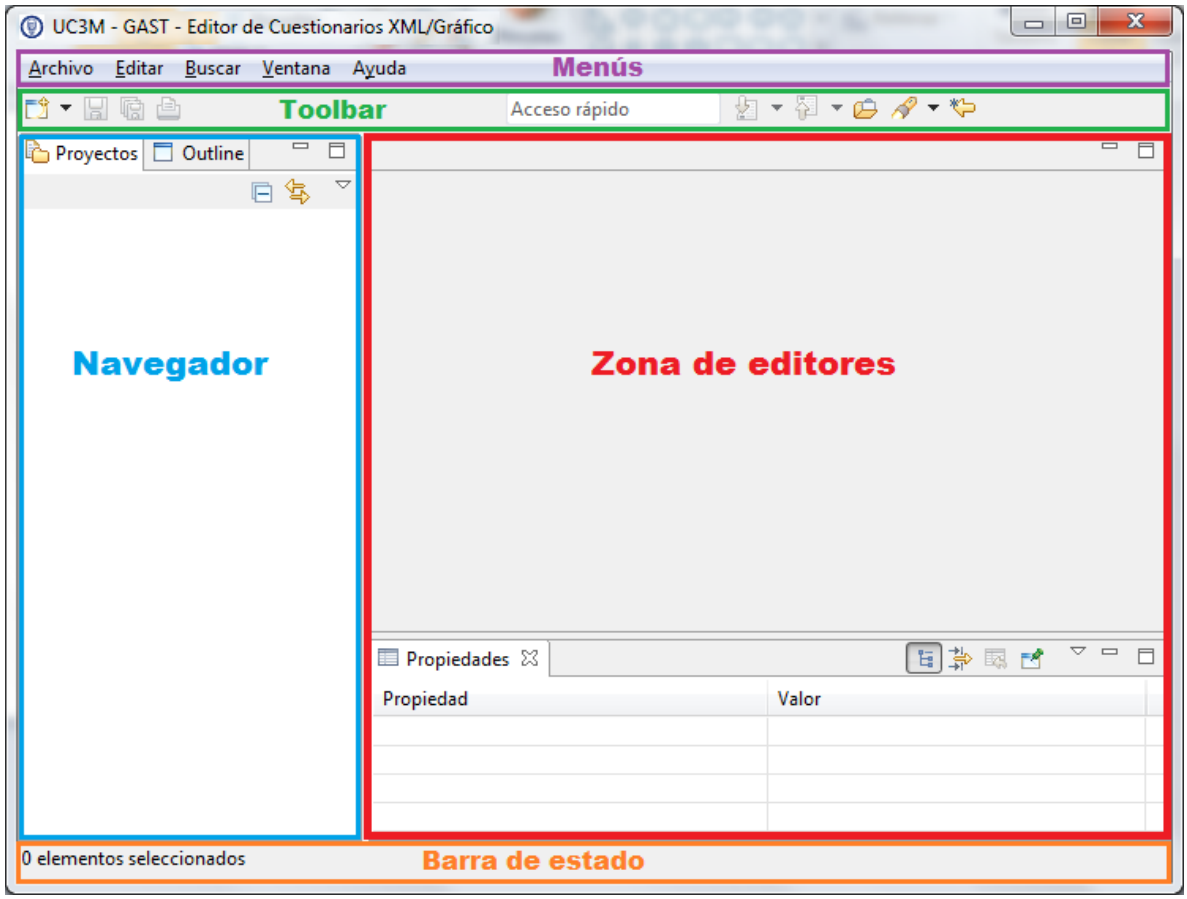


Figura 28 – Zonas de la aplicación gráfica

4.3 Creación de un proyecto de test

Una vez iniciada la aplicación y configurado el espacio de trabajo, el siguiente paso será crear un proyecto, para después añadir subcarpetas dentro de éste y ahora ya sí, poder definir ficheros de test, que es el fin de esta aplicación.

Para ello, la aplicación proporciona al usuario opciones avanzadas a la hora de crear estos ficheros, como son una serie de plantillas con diferentes configuraciones de test (incluyendo test con preguntas multi-respuesta, de respuesta simple o de texto libre). Además, es posible crear un proyecto de test ya con una estructura dada de carpetas, ficheros y recursos como imágenes, audios o videos, para facilitar al usuario el aprendizaje del sistema y hacer que estas acciones sean lo más interactivas y automáticas posibles.

Los pasos para crear un proyecto de test serían:

- **Paso 1**: crear un proyecto mediante las opciones proporcionadas por la aplicación. Estas están disponibles en el menú '*Archivo > Nuevo*', en la opción 'Nuevo' del menú contextual de la vista de proyectos, y también en la barra de herramientas bajo el menú principal.

- **Paso 2**: elegir el tipo de proyecto a crear. Como se ha dicho, es posible crear un proyecto vacío, o bien elegir un proyecto que ya cuente con una estructura de carpetas, ficheros de test y recursos gráficos por defecto, seleccionando de una lista el tipo de proyecto a crear dentro del diálogo.

Si se opta por un proyecto con una estructura ya predefinida y unos recursos por defecto, podrá seleccionarse cualquiera de los ficheros de test y ver que el editor multi-página se abre correctamente.

Si por el contrario se crea un proyecto vacío, se deberá continuar con los siguientes pasos:

- **Paso 3**: creación de directorios dentro del proyecto, a modo de organización según las necesidades (ficheros de test, recursos gráficos, etc.). Esta opción puede ser seleccionada en las mismas ubicaciones descritas en el primer paso.

- **Paso 4**: una vez creadas las carpetas, el siguiente paso sería añadir los recursos gráficos que vayan a ser utilizados en los ficheros de test. Pueden ser imágenes, audios o videos, y preferiblemente dentro de las carpetas creadas anteriormente.

- **Paso 5**: por último, y lo más importante, sería la creación de los ficheros de test. Esta opción se encuentra también en las ubicaciones descritas en el paso 1. El diálogo para crear ficheros de tipo test consta de dos páginas:

    a. La primera página permite seleccionar la ubicación exacta de dónde se desea crear el fichero, e introducir el nombre este. La extensión del fichero deberá ser obligatoriamente '.xml', y de no ser así, el sistema se lo asignará automáticamente. Si estos datos no son válidos, no se podrá acceder a la segunda página del diálogo.

    b. La segunda página requiere que se introduzca el título del test a crear, siendo posible asignarle una descripción, y dando la opción de seleccionar de una lista una plantilla por defecto para el fichero a crear.

Además, existen otras dos opciones para agregar un nuevo proyecto al espacio de trabajo:

a. La primera de ellas es mediante el uso de la opción '*Import*', que consiste en seleccionar un proyecto desde cualquier ubicación del equipo y hacer una copia local en el espacio de trabajo, o bien evitar esto y tener una referencia a ese proyecto importado, con los peligros que esto conlleva si se modifica desde dos lugar distintos o si se elimina desde uno de ellos. Por ello se aconseja realizar una copia local siempre en el espacio de trabajo. Esta función se complementa con la opción '*Export*', de forma que un proyecto puede ser exportado a cualquier otra ubicación del equipo, sin necesidad de que se trate de un espacio de trabajo diferente.

b. La segunda y más compleja, tiene que ver con la posibilidad que ofrece el sistema de compartir los proyectos creados y generar un control de versiones sobre estos. Todo ello gracias a que se permite configurar, mediante CVS o SVN, un repositorio remoto en el cual almacenar los proyectos y así poder acceder a ellos de forma segura y desde cualquier ubicación o equipo. Estas opciones serán útiles a la hora de crear un sistema a nivel superior, que utilice los proyectos creados en esta aplicación para mostrarlos en otros entornos. El inconveniente de todo esto, es que solo usuarios experimentados se atreverán a configurar dicha funcionalidad.

Estas opciones y los diálogos se detallan en el apartado anexo 8.1.4.

Una vez creado el proyecto, con su estructura y los ficheros de test (al menos uno), se podrá empezar a trabajar con el editor.

4.4 Edición de un fichero de test

Los ficheros de test se encuentran en la vista de 'Proyectos', bajo la estructura de los estos, y mediante un simple doble click o mediante la opción 'Abrir' del menú contextual, dicho contenido se mostrará en el editor multi-página.

El editor está compuesto por tres páginas: diseño, árbol y código fuente. Cualquier modificación que se efectúe en una de ellas, se verá replicado en las otros dos de manera automática. Por lo cual se recomienda ser cuidadoso a la hora de modificar el contenido del fichero manualmente, mediante las páginas de árbol o código fuente, ya que cualquier cambio no coherente a la estructura que debe tener el fichero XML, definido por su esquema, hará que las demás páginas no muestren el contenido del fichero correctamente o incluso muestren error. Se recomienda por tanto usar el editor gráfico de diseño, el cual es más intuitivo y fácil de usar, además de mostrar cómo más o menos quedaría la estructura del test definido.

A continuación se muestra un ejemplo de cómo un mismo contenido se visualiza en cada una de las páginas del editor.

Diseño

Se trata del editor gráfico de pre-visualización del test. Siempre y cuando el contenido de éste sea correcto, aparecerá la estructura de secciones, preguntas y respuestas, con las respectivas imágenes si éstas estuviesen definidas en el fichero; en caso contrario, si el fichero no fuese correcto, se mostraría en la cabecera del editor un mensaje de error.

El aspecto de éste se muestra a continuación:

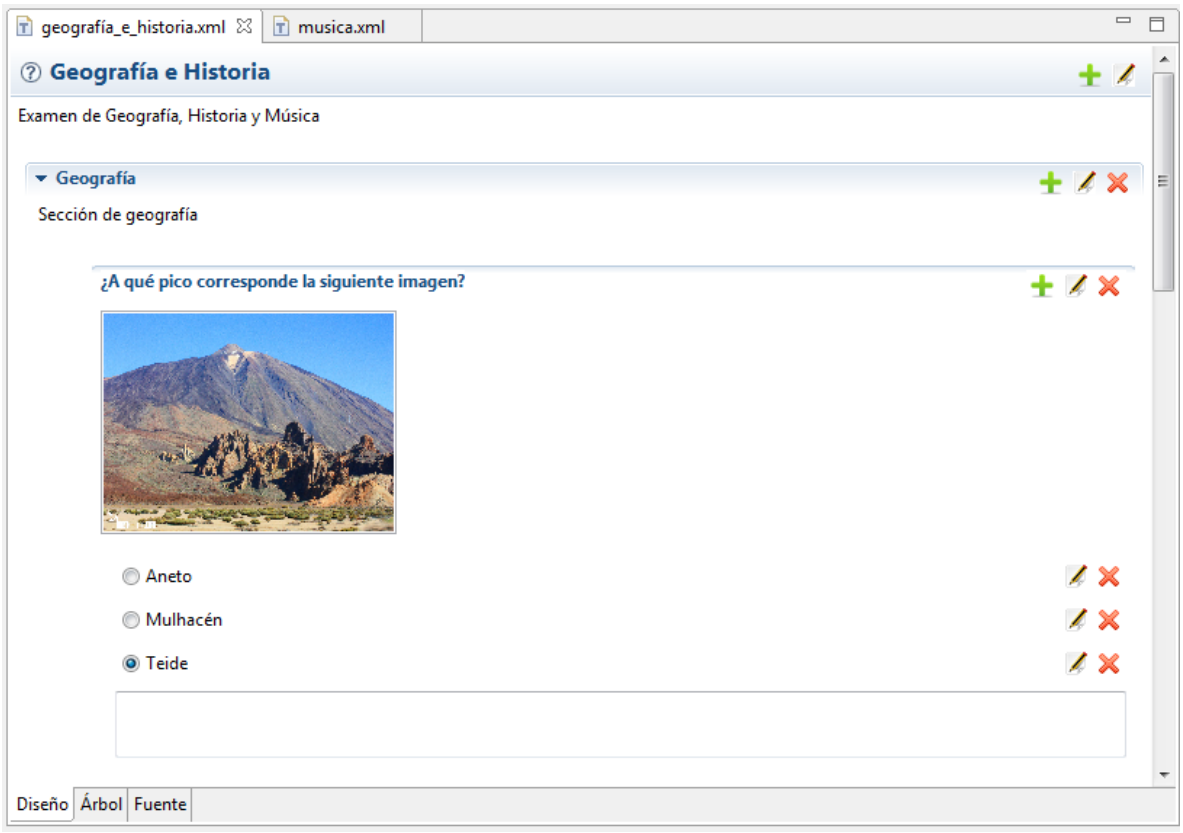


Figura 29 – Página Diseño del editor de test multi-página

El editor está desarrollado mediante Eclipse Forms, lo que le da un aspecto de web, con el título del test visible en la parte superior, permite maximizar y minimizar el contenido de las secciones para así facilitar el manejo del fichero y poder moverse con mayor facilidad por el contenido de este. Cuenta también con opciones incrustadas en cada uno de los elementos mostrados (título, secciones y preguntas), que permiten añadir elementos o modificar estos de manera rápida y sencilla, mediante sus diálogos de edición.

Dichos diálogos muestran las propiedades del elemento en cuestión, con una lista que permite modificar recursos gráficos (imágenes, audios y videos) incluidos en ese componente, y también un árbol con la estructura de elementos que hay por debajo del cual se está editando. La siguiente imagen muestra uno de esos diálogos con dichas opciones.

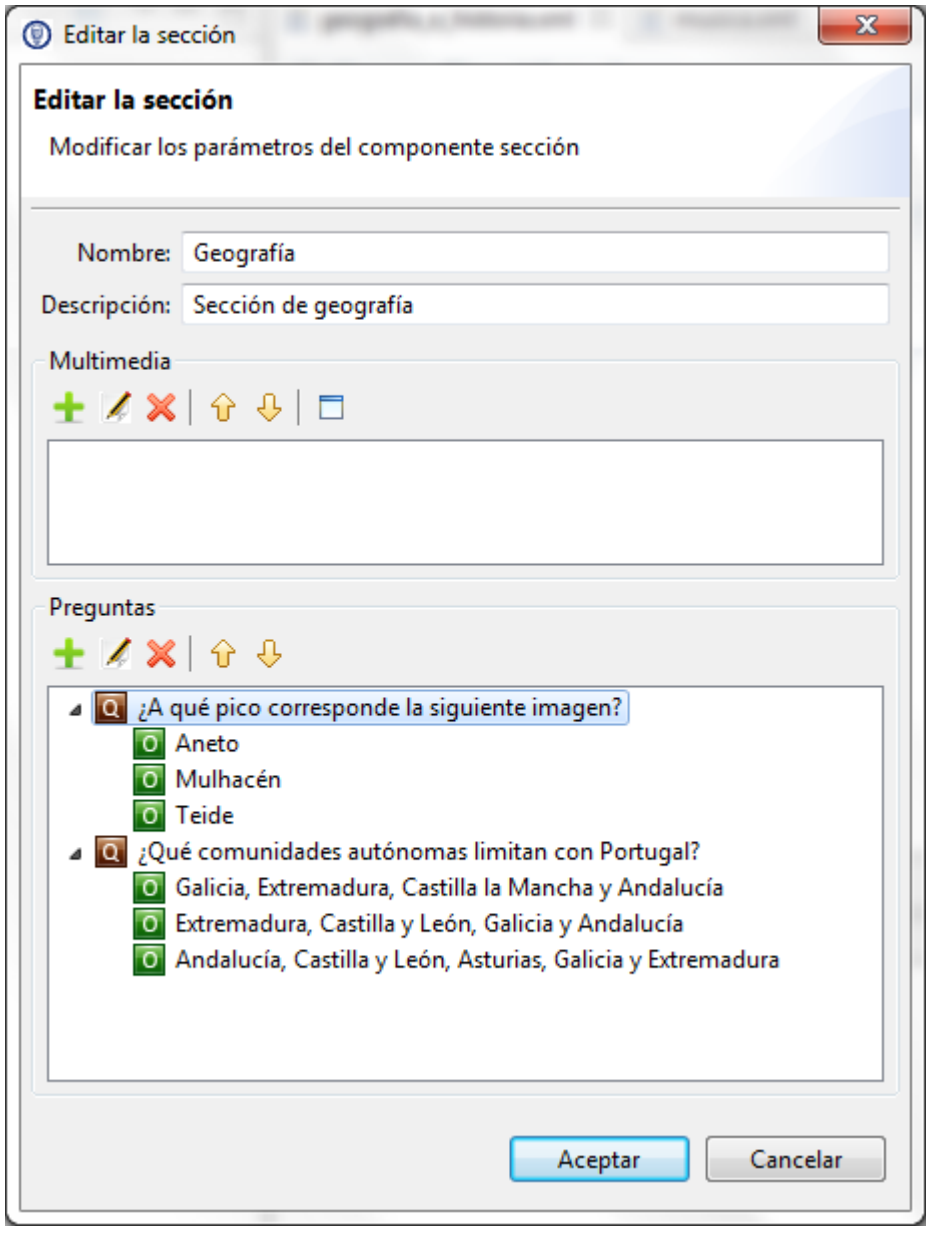


Figura 30 – Diálogo de edición de una sección del fichero de test

Además estos diálogos dan la posibilidad de modificar las propiedades de un elemento gráfico definido en el componente. Esto se hace mediante otro diálogo que permite elegir el tipo de recurso de entre imagen, audio o video, y que también da la opción de bien, elegir un recurso de la web, sin necesidad de

tener el propio recurso en la estructura del proyecto (esto incluiría tener acceso a la red para poder pre-visualizarlo), o bien seleccionar un recurso de la estructura del proyecto.

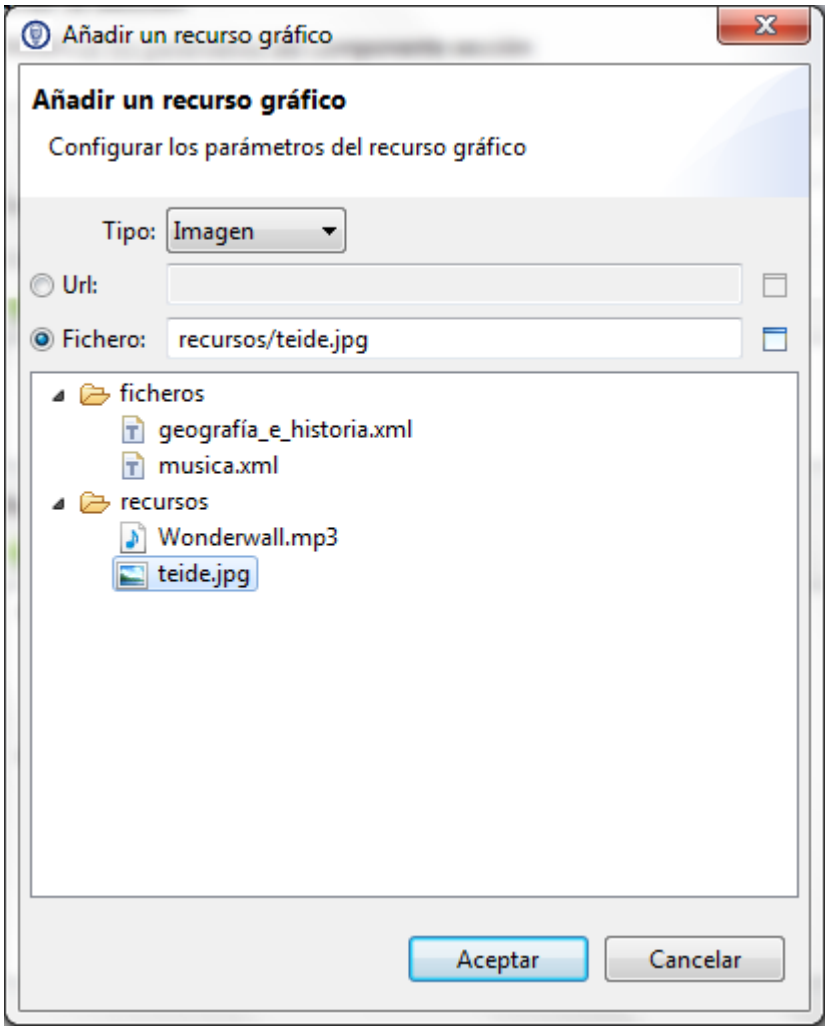


Figura 31 – Diálogo de edición de un recurso gráfico del test

En la sección 8.1.5.3 del manual de usuario puede ver más información acerca de esta página del editor.

Árbol

La segunda de las páginas del editor muestra en forma de árbol la estructura de nodos que componen el fichero XML.

En la imagen disponible a continuación se muestra su apariencia:

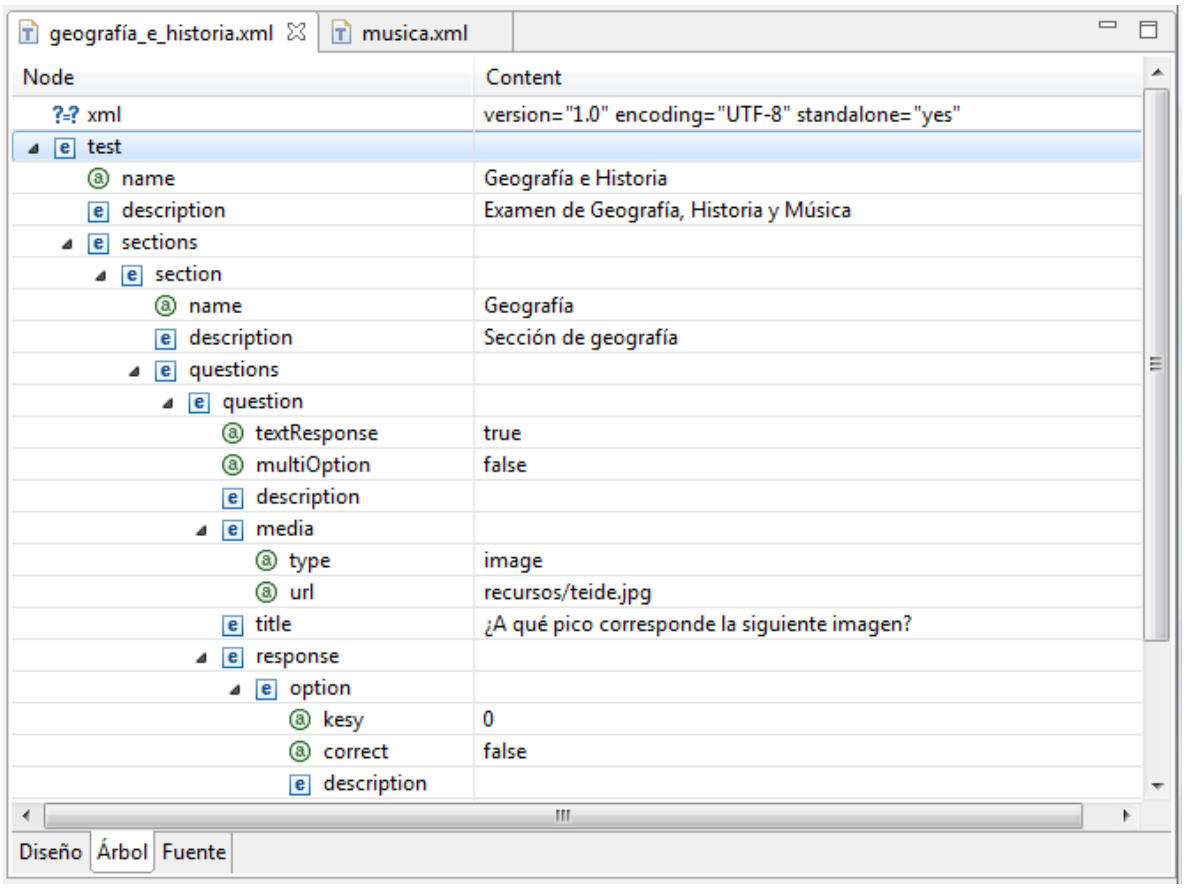


Figura 32 – Página Árbol del editor de test multi-página

La vista permite hacer modificaciones sobre el contenido del fichero, de manera que es posible cambiar directamente el valor de los atributos en la tabla. Dispone de un menú contextual con opciones que dan la oportunidad de eliminar o reemplazar el elemento seleccionado, así como de añadir nuevos atributos, elementos, antes o después de este.

Además es posible arrastrar y soltar elementos de un lado a otro de la tabla. Sin embargo no es aconsejable hacer uso de esta opción a no ser que se tenga conocimiento de qué se está moviendo y a donde, ya que el fichero podría sufrir modificaciones que no cumplan con la estructura lógica del test. En ese caso, la página de *Diseño* mostraría un error y no podría ser utilizada hasta que el formato del fichero volviese a ser correcto.

En la sección 8.1.5.2 del manual de usuario puede ver más información acerca de esta página del editor.

Fuente

Finalmente, la última de las páginas del editor mostrará el contenido del fichero en formato XML. Se trata de un editor de texto común, que muestra el contenido del fichero de manera estructurada y fácil de comprender.

Esta página tendrá la apariencia mostrada a continuación:

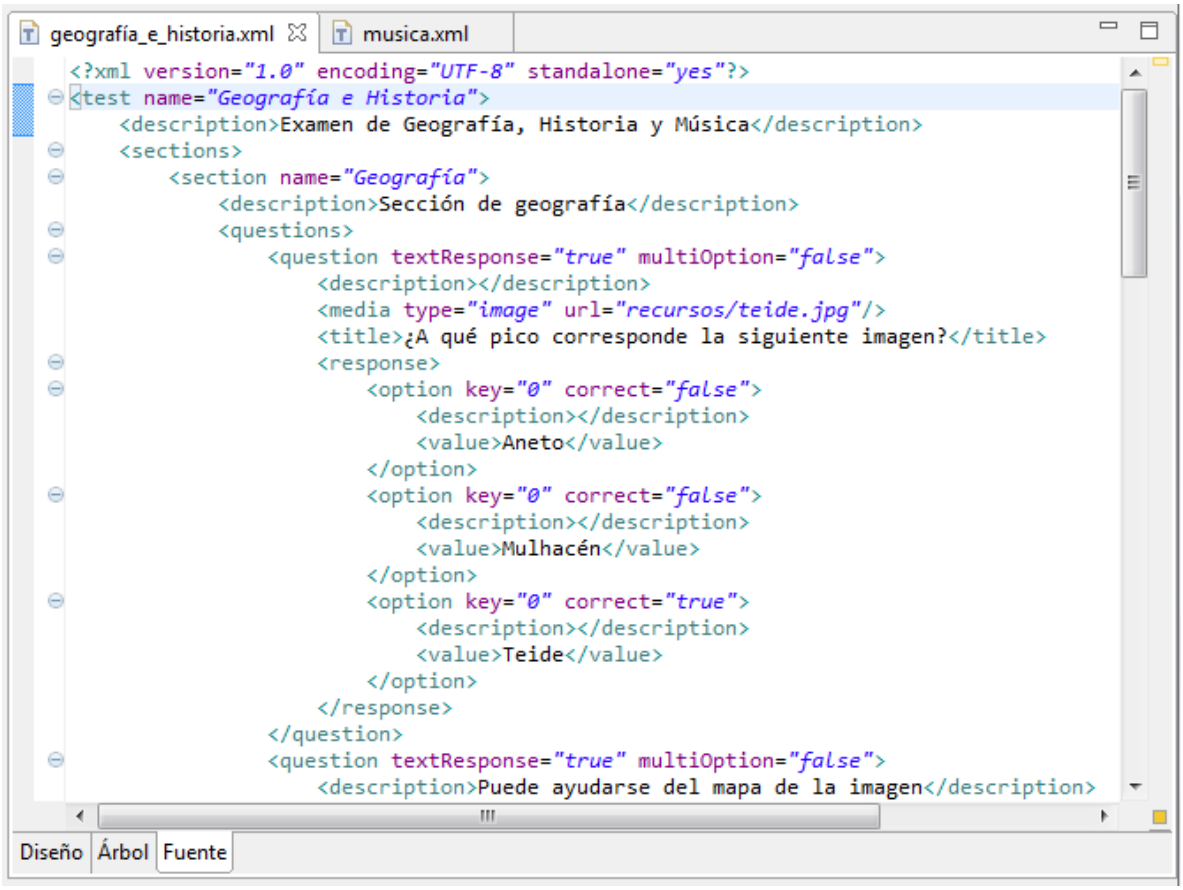


Figura 33 – Página Fuente del editor de test multi-página

Si se desea editar el contenido del fichero mediante esta vista, es posible usar la ayuda contextual (teclas de 'Control '+'Espacio') que muestra los posibles elementos y atributos que se pueden escribir en esa parte del fichero. Esta opción es útil para usuarios expertos en edición de XML, aun que como se ha dicho anteriormente, se recomienda usar el editor gráfico.

Dispone de un menú contextual con en gran número de opciones de edición, como pueden ser revertir cambios, cortar, copiar o eliminar la selección, comentar esta automáticamente, validar o formatear el contenido…, y otras opciones que permiten gestionar el fichero, como pueden ser la de imprimirlo, compartirlo con un repositorio de contenidos, compararlo o volver a una versión anterior, etc.

Además, si el contenido mostrado no es correcto y tiene algún error, aparecerá una señal (o muesca) roja (o amarilla en caso de error menor) en los laterales del editor. Si se sitúa el cursor del ratón sobre esta, se mostrará un mensaje con la causa de dicho problema.

En la sección 8.1.5.1 del manual de usuario puede ver más información acerca de esta página del editor.

Capítulo 5

Gestión y planificación del proyecto

5.1 Historia del proyecto

El proyecto ha pasado por distintas etapas hasta llegar a la solución implementada. Se ha modificado el enfoque durante el transcurso del mismo y a continuación se detallan los cambios que se han realizado hasta llegar a la solución final.

El proyecto original se trataba de una página web, que se encargaba de mostrar los resultados proporcionados por otra aplicación en la cual se formulaban una serie de preguntas y los resultados eran enviados a una base de datos, de la cual la aplicación web a implementar los extraería y los representaría según qué tipo de usuario. Esta web se basaba en el e-Learning, al igual que la solución final implementada.

La implementación de esta web se llegó iniciar, se desarrolló una pequeña maqueta, en la cual ya se representaba información interesante. Se estaba realizando bajo la distribución web de Eclipse RCP, llamada Eclipse RAP, que proporciona herramientas para realizar de manera sencilla una página web en Java. Se apoyada en Hibernate para favorecer el acceso a las bases de datos y obtener objetos también Java. Ambas tecnologías, Eclipse RAP e Hibernate, fueron investigadas por el desarrollador exclusivamente para el desarrollo de dicha aplicación, y se llegó a obtener un buen resultado y a sacarles partido a ambas.

El rumbo del proyecto cambió cuando se analizó la idea de implementar un editor de ficheros de test, desarrollarlo en primera instancia bajo Eclipse RCP, para posteriormente, con algunas modificaciones, incluirlo en la aplicación web. Esto es posible gracias a que Eclipse RCP y Eclipse RAP pueden llegar a funcionar de manera muy similar, a pesar de estar diseñados para fines completamente diferentes.

Finalmente, esta idea no se pudo llevar a cabo, debido a las limitaciones que aún tenía Eclipse RAP, pero sobre todo fue debido a que el propio editor de test desarrollado para este proyecto se ha convertido por si solo en un proyecto de gran envergadura.

El aprendizaje de Eclipse RCP para sacarle el mayor partido posible a sus opciones, con todas las posibilidades que ofrece, ya ha supuesto un gran esfuerzo. A este aprendizaje se debe sumar también el diseño de la aplicación, creando una herramienta con módulos independientes y cada uno de estos con entidad propia, pudiendo ser reutilizados en cualquier otra implementación. Cabe destacar además que ha sido necesario realizar un estudio de aplicaciones similares con el fin de sacar conclusiones de cara a mejorarlas.

5.2 Ciclo de vida

Para conseguir la solución que se expone en esta memoria, se ha mencionado anteriormente que antes de comenzar el desarrollo de lo que ha sido este proyecto, existió una etapa en la cual se comenzó a implementar una herramienta que finalmente no ha formado parte de este proyecto. Por este motivo, no se tendrá en cuenta en los siguientes puntos, centrando únicamente la atención en lo que es realmente la aplicación que se ha desarrollado. La implementación de la solución se ha llevado a cabo en una serie de etapas, las cuales se describen a continuación:

- **Fase de inicio:** Durante esta etapa se ha realizado un estudio de herramientas que proporcionen soluciones similares a la que busca conseguir esta aplicación, para posteriormente pasar a definir los requisitos funcionales y no funcionales que hagan que esta herramienta se diferencie del resto.

- **Fase de preparación:** Una vez definidos los requisitos, en esta fase se ha procedido a realizar un estudio para buscar las diferentes tecnologías que podrían ser usadas a la hora de implementar la aplicación. Se ha tratado de buscar aquellas que proporcionen una solución fiable y robusta a la hora de crear el entorno gráfico, de manera que se cubran los requisitos previamente definidos, así como de encontrar el entorno de desarrollo que permita trabajar con todas ellas de manera simultánea.

- **Fase de desarrollo:** Una vez que se ha tomado la decisión de qué tecnologías se van a usar, se ha procedido a iniciar el desarrollo de la aplicación. Durante este tiempo se ha ido pasando por diferentes etapas, en las cuales se han ido incluyendo funcionalidades que cubriesen los requisitos marcados al inicio. Se trata de la etapa más extensa en cuanto a tiempo y dedicación se refiere.

- **Fase de finalización:** Es el periodo en el cual se ha verificado el correcto funcionamiento de la aplicación. En esta fase se comprueba si se han cumplido los requisitos marcados al inicio, también esta fase se solucionan aquellos problemas que hayan podido surgir con las pruebas de la aplicación y se implementan las mejoras oportunas que aparezcan durante las pruebas. Además, en esta fase se ha finalizado con el desarrollo de la memoria.

A continuación se muestra un gráfico con los periodos por los cuales ha transcurrido el proyecto. En el eje horizontal se define la variable tiempo y en el eje vertical se define el esfuerzo dedicado en horas/Semana en base a las semanas de trabajo empleado y el esfuerzo llevado a cabo en cada periodo de tiempo.

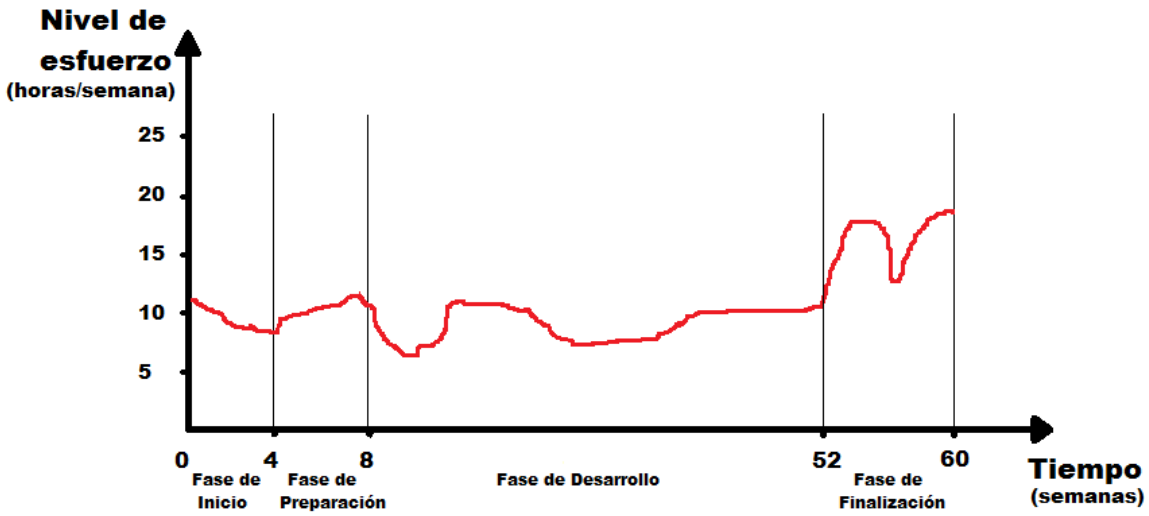


Figura 34 – Gráfica nivel de esfuerzo / tiempo

5.3 Organización

Las cuatro fases expuestas anteriormente, inicio, preparación, desarrollo y finalización, se subdividen a su vez en diferentes etapas o actividades siguiendo un orden definido.

A continuación se muestra una tabla con la descripción de cada una de estas etapas, así como de las semanas en las cuales se ha desarrollado y el tiempo empleado en cada una de ellas de una manera aproximada.

Fase	Actividad	Semanas	Horas
Inicio	Estudio de herramientas similares	1 y 2	12
	Definición de requisitos	3 y 4	8
	Memoria	3 y 4	10
	TOTAL FASE DE INICIO	**1 a 4**	**30**
Preparación	Elección y estudio de las tecnologías a utilizar	5 a 7	20
	Búsqueda y elección de un entorno de desarrollo apropiado	7 y 8	5
	Memoria	6 a 8	15
	TOTAL FASE DE PREPARACIÓN	**5 a 8**	**40**
Desarrollo	Configuración del entorno de desarrollo e instalación de las tecnologías elegidas	9 y 10	10
	Creación de la ventana principal de la aplicación: definición del plugin *Aplicación*	11 y 12	10
	Inclusión de la primera de las vistas, el navegador de proyectos: creación y adaptación del plugin *Navegador*	13 a 16	25
	Configuración del menú superior y la toolbar de la aplicación, y sus opciones	17 y 18	20
	Estudio y creación de las interfaces que definen la estructura elegida para los ficheros de test: plugin *Estructura de Test*	19 y 20	15
	Adaptación de la estructura elegida para los test a clases Java que vuelcan su contenido sobre XML: plugin *Estructura de Test XML*	21 a 23	20

	Definición del editor XML multi-página de test: plugin *Editor Multi-página*	24 a 27	30
	Implementación de la página de diseño y pre-visualización del editor de test: plugin *Editor Gráfico*	28 a 40	80
	Configuración del menú contextual del navegador	40 a 43	35
	Añadir los diálogos de creación de proyectos y ficheros de test que contienen plantillas	44 a 46	25
	Opciones de configuración de repositorios compartidos: CVS y SVN	47	5
	Internacionalización de la aplicación: inglés y español	48 y 49	15
	Memoria	9 a 51	40
	TOTAL FASE DE DESARROLLO	**9 a 51**	**330**
Finalización	Pruebas: manuales e implementación de un plugin de pruebas unitarias	52 a 57	25
	Solución de problemas e implementación de mejoras	54 a 57	30
	Manual de la aplicación e inclusión en el menú de ayuda	57 y 58	15
	Memoria	52 a 60	60
	TOTAL FASE DE FINALIZACIÓN	**52 a 60**	**130**
	TOTAL DEL PROYECTO	**60**	**530**

Tabla 10 – Fases del proyecto

5.4 Presupuesto

Este apartado trata de mostrar en forma de números el esfuerzo y las cantidades económicas que podría suponer la realización de un proyecto con este alcance.

Se dividen dichos cálculos en costes de personal y tecnológicos, y se hace un resumen final con el total.

5.4.1 Coste de personal

Se trata del coste que supone el trabajo realizado por las personas físicas que han colaborado en el proyecto. Anteriormente se citaba que en el proyecto han participado dos personas: el alumno que ha desarrollado la aplicación, y un jefe de proyecto que se ha encargado de dirigir y orientar en la realización de la aplicación, con diferentes funciones entre las que podríamos destacar: realización de recomendaciones acerca de qué tecnologías existían ya similares a la solución propuesta, supervisión de las distintas etapas que componen el desarrollo, verificación del desarrollo realizado, verificación de la aplicación de conocimientos que se presuponían en fase inicial para obtener el mejor de los resultados posible, guía y revisión de la memoria…

A continuación se muestra la tabla de costes con los salarios que podrían percibir cada uno de ellos, determinado a partir del sueldo según su rango y a las horas dedicadas por cada uno de ellos en la realización del proyecto.

Apellidos y Nombre	Categoría	Dedicación	Coste por hombre	Coste
Fernández Panadero, Mª Carmen	Jefa de Proyecto	70 h	40 €/h	2.800 €
Domínguez Vicios, Jesús	Alumno desarrollador	530 h	20 €/h	10.600 €
TOTAL		**600 horas**		**13.400 euros**

Tabla 11 – Coste de personal

5.4.2 Coste tecnológico

Se trata del equipo utilizado para el desarrollo de la aplicación, de las posibles licencias requeridas, así como de unos gastos derivados del uso de la red eléctrica y teléfono.

Dado que el equipo utilizado para el desarrollo se trata del ordenador personal del alumno, se calculará su coste en función del total y del tiempo de uso que se estima, y así sacar un total de cuanto ha supuesto este gasto.

En cuanto a las licencias, se ha optado por utilizar todas las tecnologías y herramientas gratuitas a hora de desarrollar la aplicación, por lo que la adquisición de licencias en este sentido no ha supuesto un coste adicional. Sin embargo, la memoria se ha redactado usando las herramientas proporcionadas por el paquete Microsoft Office, concretamente su aplicación Microsoft Word como editor de texto, Microsoft Visio para elaborar los diagramas de flujo, y Microsoft Power Point para las plantillas que forman la presentación de la aplicación.

De esta forma, el coste tecnológico se muestra a continuación:

Descripción	Coste	Dedicación	Depreciación	Coste
Ordenador portátil	600 €	14 meses	60 meses	140 €
Licencia Microsoft Office	80 €			80 €
TOTAL				**220 euros**

Tabla 12 – Coste tecnológico

5.4.3 Coste total

Como conclusión, para evaluar el coste total del proyecto, éste será la suma del coste de personal y el coste tecnológico. A ello hay que añadir los costes indirectos derivados de la realización del proyecto, que se han estimado en un 20%. Además, se aplicará un IVA del 21%.

La siguiente tabla muestra el total del coste de esta aplicación.

Descripción	Coste
Coste de personal	13.400 €
Coste tecnológico	220 €
Costes indirectos (20%)	2.724 €
SUBTOTAL (sin IVA)	**16.344 euros**
IVA (21%)	3.432 €
TOTAL	**19.776 euros**

Tabla 13 – Coste total

Capítulo 6

Resultados y conclusiones

En general, los resultados obtenidos se pueden calificar como muy positivos. Se han cubierto todas las exigencias marcas al inicio de la implementación y la conclusión de todo ello es que se ha obtenido un editor de test que aporta algo nuevo y diferente a los ya existentes.

Esta afirmación está justificada de manera que la aplicación permite realizar desarrollos de proyectos de test de forma paralela por diferentes usuarios, gracias a que los proyectos generados pueden ser compartidos a través de repositorios de contenido, que a su vez dan la opción de generar un control de versiones de esos proyectos y realizar copias de seguridad fácilmente. Esto permite que un usuario pueda disponen de esos proyectos de forma remota, así como ver la evolución que un determinado proyecto de test o sus ficheros de test han ido teniendo, y así poder rescatas versiones anteriores o ver qué usuario ha realizado ciertos cambios a esos proyectos.

Otro punto de la aplicación a tener en cuenta, es su sencillez a la hora de crear nuevos proyectos de test. Mediante una interfaz sencilla se puede generar, a través de un diálogo simple y que ofrece plantillas por defecto, una estructura de test compuesta por diferentes ficheros de test. Por su parte, la edición de estos test resulta intuitiva y sencilla de comprender gracias al editor gráfico proporcionado, donde mediante simples diálogos se pueden añadir nuevos parámetros a esos test o modificar los ya existentes. Además la interfaz se acompaña de una opción que permite visualizar un manual de usuario, donde de forma estructurada se explican todas las opciones proporcionadas por la aplicación.

Además, la implementación mediante Eclipse RCP aporta al proyecto una serie de cualidades importantes a tener en cuenta, como son la modularidad de la aplicación, en la que cada una de las partes es independiente, lo que permite ser reutilizada o sustituida fácilmente por otra que aporte algo nuevo. Además, esta tecnología es robusta y se encuentra en continuo desarrollo,

A continuación se justifica cómo y en que se ha basado el desarrollo para dar cobertura a cada uno de los requisitos funcionales:

- Como se dice, la interfaz gráfica es simple, las opciones son sencillas de entender y fáciles de encontrar. Mediante pocos pasos,

concretamente en solo tres o cuatro clics de ratón y un nombre, se puede crear un nuevo proyecto o un nuevo fichero de test. Además, el editor gráfico es bastante intuitivo a la hora de modificar esos ficheros de test de manera rápida y eficiente.

- No precisa de conexión a la red para poder utilizar la aplicación, únicamente cuando se desee compartir o importar proyectos desde un repositorio de contenidos remoto.

- El formato definido para los test es sencillo, ya que aunque se trate de formato XML, la estructura de es bastante simple, con un número reducido de elementos y atributos. Además, mediante el analizador implementado sería fácil reusar estos ficheros XML en otros desarrollos.

- Se han añadido opciones que permiten configurar repositorios de contenido compartidos, concretamente los dos más usados: CVS y Subversión. Con ello se da soporte a la posibilidad de disponer de un control de versiones de los proyectos o ficheros de test.

- Otra de las cualidades de Eclipse RCP, es que permite generar versiones de la aplicación para diferentes sistemas operativos. También la implementación se ha diseñado de tal manera que es posible generar la aplicación en diferentes idiomas (español e inglés), y resultaría sencillo configurar cualquier otro idioma.

En cuanto a los requisitos no funcionales:

- El desarrollo se ha estructurado en diferentes plugins, dependiendo de su funcionalidad. De esta forma se cubre con la premisa de disponer de un sistema modular.

- Gracias a esto último, se ha conseguido que cada una de las partes que forman la aplicación pueda ser reutilizada.

- Se ha pensado en todo momento en las funcionalidades futuras que podrían derivar del diseño y desarrollo de este proyecto, de manera que el formato de los test es independiente y puede ser modificado y exportado a otros sistemas.

- En cuanto a la calidad del código generado, se ha desarrollado usando las herramientas de codificación proporcionadas por el entorno de desarrollo Eclipse, de manera que el código es legible, estructurado, con una tasa de comentarios aceptable y cumple con las reglas de calidad definidas.

A modo de opinión personal, añadir que el desarrollo no ha resultado complicado, sino más bien laborioso. Se han aprendido nuevas tecnologías y eso

siempre resulta complicado. Es cierto que han aparecido problemas que en ocasiones han supuesto muchas horas de trabajo y esfuerzo, que no deberían ser calificadas como tiempo "perdido", sino más bien como tiempo dedicado a la formación como desarrollador.

De ello se sacan siempre conclusiones positivas, y este es el objetivo real del proyecto, descubrir nuevas tecnologías, enfrentarse a un desarrollo desde cero, buscar una idea innovadora que aporte algo nuevo, definir unos objetivos, y sobre todo, aprender día a día de los problemas que van apareciendo mientras se implementa la solución escogida.

Capítulo 7

Trabajos futuros

Son muchos los caminos que puede tomar la continuación a este proyecto, y se pueden ver desde diferentes puntos de vista.

Por un lado, se podría mejorar la estructura de los ficheros de test que define la aplicación, ya sea introduciendo nuevas funcionalidades o mejorando las ya implementadas. Se podrían definir nuevos tipos de pregunta, llegando incluso a plantearse la posibilidad de dar soporte a los tipos de ficheros SCORM. Se habla a continuación de esta posibilidad, centrándose en los cambios que tendrían que realizarse en la aplicación.

Siguiendo esta línea de mejoras, se podría dar la posibilidad de definir ficheros de test única y exclusivamente con contenido de preguntas, con sus respuestas. De esta forma, se podría crear una pila de preguntas y ser reutilizadas para crear ficheros de test más complejos.

Desde el punto de vista del entorno gráfico, se podrían añadir nuevas opciones a los menús. Además, se podrían crear nuevas vistas que faciliten el uso del editor, configurar ciertas opciones para que puedan ser reusadas

Si se toma como punto de vista para ofrecer mejoras que uno de los requisitos era crear un editor de test que fuera independiente de cómo luego se iban a usar los proyectos creados en este, se podría continuar, desarrollando ese otro u otros proyectos que recojan los datos creados por esta aplicación, por ejemplo de un repositorio compartido, y usándolo para crear un test interactivo en una Tablet Android o en una web.

Estos son solo algunos puntos hacia donde podría derivar este proyecto. A continuación se exponen específicamente cada uno de ellos.

7.1 Adaptación del editor a SCORM

Para poder realizar esta adaptación, lo primero sería replantear el modelo de test definido en la aplicación, de modo que se debería dar cabida a las numerosas características que conforman SCORM y a su amplio abanico de tipo de preguntas y sobre todo respuestas que contiene.

También sería necesario, crear un nuevo o modificar el editor gráfico generado para este proyecto, de manera que sea capaz de representar todas y cada una de los tipos de respuesta que SCORM contiene, así como cambiar el modelo lógico que crea gráficamente la estructura de test.

El paso a soportar este tipo de ficheros sería muy importante a la hora de que los portales que ya soportan SCORM pudiesen contar con este editor gráfico.

Sin embargo, el trabajo no sería sencillo, debido a que SCORM es un modelo de test muy avanzado, con multitud de opciones, y por tanto habría que replantear muchas de las características de este sistema.

7.2 Modificación de los ficheros de test

La modificación de la estructura de los ficheros de test, puede derivar en la implementación de importantes mejoras.

Por un lado, la posibilidad de definir preguntas sin necesidad de definir la estructura completa de un test y sus secciones. Esta opción añadiría un conjunto de posibilidades a la hora de reutilizar las preguntas y generar ficheros de test con mayor complejidad. Así, se podría crear una pila de preguntas, agruparlas en función de distintos criterios, como su género, tipo de respuesta, etc.

Por otro lado, a los ficheros de test se les podrían incluir mejoras, tales como la posibilidad de disponer de nuevos tipo de respuesta. Esto puede ir de la mano de la adaptación a SCORM, definida anteriormente, o simplemente definir otros tipos de respuesta que puedan ser potencialmente requeridas por los usuarios.

Para llevar a cabo estas mejoras, se debe proceder a la modificación de la lógica de los ficheros de test desarrollada hasta ahora, tanto en su estructura (interfaz y XML) como en el editor gráfico, añadiendo las nuevas opciones al formato definido e implementando estas mejoras para poder ser representadas gráficamente en el editor.

Adicionalmente, habría que verificar que la versión del editor gráfico actual sigue funcionando, independientemente del tipo de fichero de test y su estructura.

7.3 Ampliación y mejora de la aplicación

Añadir nuevas funciones a la aplicación, como por ejemplo:

- Añadir el diálogo de Preferencias, que permitiese definir los parámetros del editor XML, cambiar los colores de este, etc.

- Crear una vista similar a la denominada 'Esquema' (u Outline), la cual muestra la estructura del fichero XML en forma de árbol. Dicha vista puede mostrar la estructura del fichero de test, pero utilizando el árbol empleado en los diálogos de edición de elemento (secciones, preguntas y respuestas). De esta forma, se podrían añadir directamente sobre dicha vista las opciones de crear, modificar o eliminar componentes, como se hace en el diálogo, o también añadir otro tipo de opciones, como la de arrastrar y soltar elementos.

Por otro lado se pueden mejorar las funcionalidades ya existentes, como por ejemplo conseguir que las opciones del menú de edición funcionen también sobre la página del editor gráfico, y no únicamente sobre el editor de código fuente XML.

7.4 Aplicación de nivel superior

Algo que sí sería factible de llevar a cabo y que se trata de uno de los principales objetivos de este proyecto, (de dónde surgió la idea) sería la implementación del siguiente nivel dentro del ciclo generación-representación/respuesta-resultados.

Una vez se dispone de la aplicación que genera los ficheros de test, estos pueden ser compartidos mediante un repositorio de contenidos, y usados por una segunda aplicación a modo de representación, que de la posibilidad a usuarios de contestar a las preguntas de los test generados.

Estas aplicaciones podrían ser una página web, una aplicación Android, etc. Estas se conectarían al repositorio de contenidos y obtendrían los proyectos de test generados y compartidos mediante el editor. De esta forma, se podría reutilizar el plugin que define la estructura de los ficheros de test en formato XML para así convertirlos a clases Java, y de esta forma usar esos objetos para representar gráficamente los test según el tipo de tecnología utilizada.

7.5 Otras líneas futuras

Se podría añadir al formato de test predefinido soporte a maquetación, mediante el uso de CSS. Sería para poder mejorar el editor gráfico, ya que se podrían añadir formatos de texto, colores y otro tipo de herramientas que mejorasen la pre-visualización de los ficheros de test.

Bibliografía

[1] Definición de e-Learning

http://www.e-abclearning.com/definicione-learning

[2] Definición de MOOCs

https://es.wikipedia.org/wiki/Mooc

[3] Definición de SCORM

http://www.2teach.es/2012/11/moodle-y-scorm/#.VhvyYPkqyJg

[4] Definición de QTI

http://es.unionpedia.org/QTI

[5] Introducción a Moodle

https://docs.moodle.org/all/es/Acerca_de_Moodle

[6] Módulo de cuestionarios en Moodle

https://es.wikipedia.org/wiki/Moodle#M.C3.B3dulos_principales_en_Moodle

[7] Introducción a Dokeos

http://plataformadokeos.blogspot.fr/

[8] Utilidad de test de Dokeos

https://www.dokeos.com/produce-tests-29-types-questions-possible-dokeos/

[9] Introducción a eXe Learning

http://exelearning.net/

[10] Introducción a Hot Potatoes

http://www.ite.educacion.es/formacion/materiales/62/cd/modulo_4_tipos_de_ejercicios/index.html

[11] Características de Hot Potatoes

http://www.ite.educacion.es/formacion/materiales/62/cd/modulo_1_primeros_pasos/index.html

[12] Información de Hot Potatoes

https://hotpot.uvic.ca/

[13] Definición de XML y Schema

http://www.hipertexto.info/documentos/xml.htm

[14] Usos de XML

http://www.itespresso.es/el-xml-podria-dotar-a-internet-de-inteligencia-26235.html

[15] Definición de DTD

http://www.mclibre.org/consultar/xml/lecciones/xml_dtd.html

[16] Introducción a Java

http://definicion.de/java/

[17] Versiones de Java

https://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

[18] Introducción a Eclipse

http://www.genbetadev.com/herramientas/eclipse-ide

[19] Importancia de Eclipse

http://ymasjava.blogspot.fr/2012/02/eclipse-rcp-tutorial-lars-vogel-version.html

[20] Definición de Eclipse

https://es.wikipedia.org/wiki/Eclipse_%28software%29

[21] Definición de plugin

http://ymasjava.blogspot.fr/

[22] Introducción de Eclipse RCP

http://www.avanet.org/aprendiendo-rcp-rich-client-platform-1a-parte.aspx

[23] Estructura de un plugin de Eclipse

http://www.elclubdelprogramador.com/2012/05/21/ide-construyamos-un-plug-in-de-eclipse-paso-a-paso/

[24] Definición de SWT

http://zetcode.com/gui/javaswt/

[25] Definición de widget

http://www.alegsa.com.ar/Dic/widget.php

[26] Introducción a SWT

https://es.wikipedia.org/wiki/SWT

[27] Definición de JFace

http://blog.espol.edu.ec/machavez/2012/09/09/swt-el-standard-widget-toolkit/

[28] Paquetes de JFace

http://www.javaworld.com/article/2074837/core-java/rich-clients-with-the-swt-and-jface.html

[29] Definición de Eclipse Forms

https://eclipse.org/articles/Article-Forms/article.html

[30] Introducción a CVS y Subversion

http://www.iiia.csic.es/udt/es/blog/jrodriguez/2008/subversion-vs-cvs-guia-rapida-subversion-svn

[31] Definición de CheckStyle

http://tratandodeentenderlo.blogspot.fr/2009/10/herramientas-de-analisis-de-calidad-del.html

[32] Características de Netbeans

https://es.wikipedia.org/wiki/NetBeans

[33] Definición de Netbeans RCP

http://www.javahispano.org/antiguo_javahispano_org/2004/4/17/netbeans-platform-vs-eclipse-rcp.html

[34] Características de Swing

https://es.wikipedia.org/wiki/Swing_%28biblioteca_gr%C3%A1fica%29

Capítulo 8

Anexos

8.1 Manual de Usuario de la Aplicación

8.1.1 Introducción

Se trata de una aplicación de escritorio que permite la creación y edición de ficheros en formato XML. Este tipo de fichero sigue una estructura lógica predefinida que a su vez define las partes del cuestionario.

Este manual explica cómo usar la aplicación: configuración, ciclo de vida, uso de todas sus opciones, creación y gestión de los proyectos de test, creación y edición de los propios ficheros de test, etc.

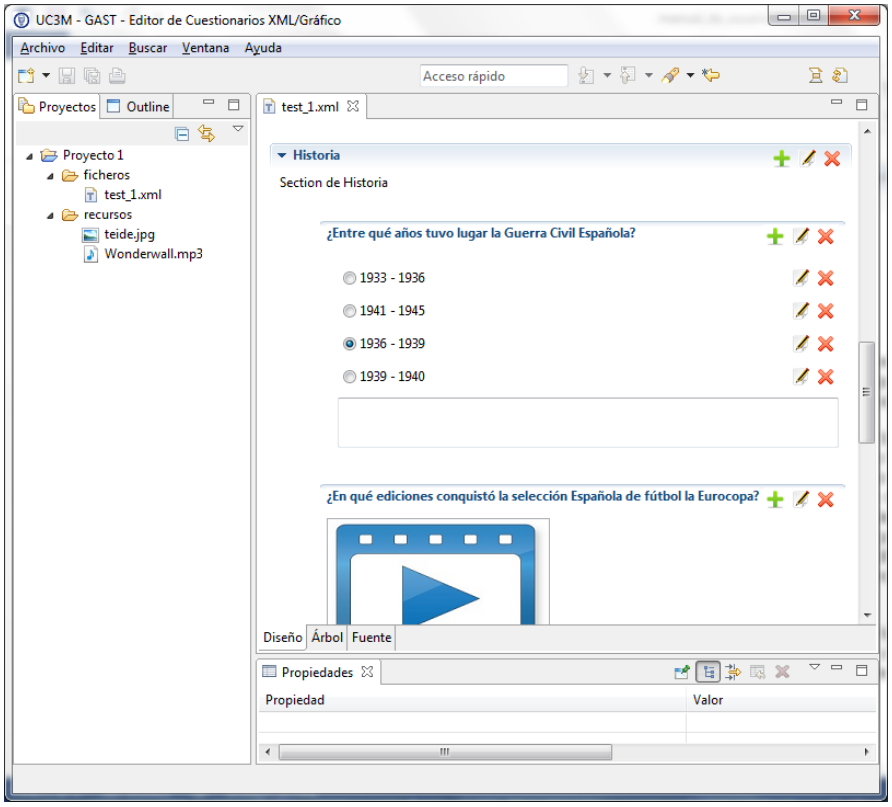


Figura 35 – Vista de la aplicación

El entorno se ha desarrollado bajo Eclipse RCP, incluyendo multitud de opciones proporcionadas por dicha tecnología. En concreto, para la edición de cuestionarios se ha implementado un editor multi-página, que permite editar los ficheros XML en modo texto, en modo de árbol (modificando los valores, moviendo las etiquetas, etc.), o bien mediante la opción gráfica, la cual incorpora un pre-visualizador que permite intuir cómo se representarían los cuestionarios. Además, el editor gráfico permite añadir, modificar o eliminar partes del cuestionario de manera rápida e intuitiva, sin necesidad de comprender su formato XML.

8.1.2 Ciclo de vida

Se trata de los pasos a dar para iniciar la aplicación, configurarla y controlar su ciclo de vida.

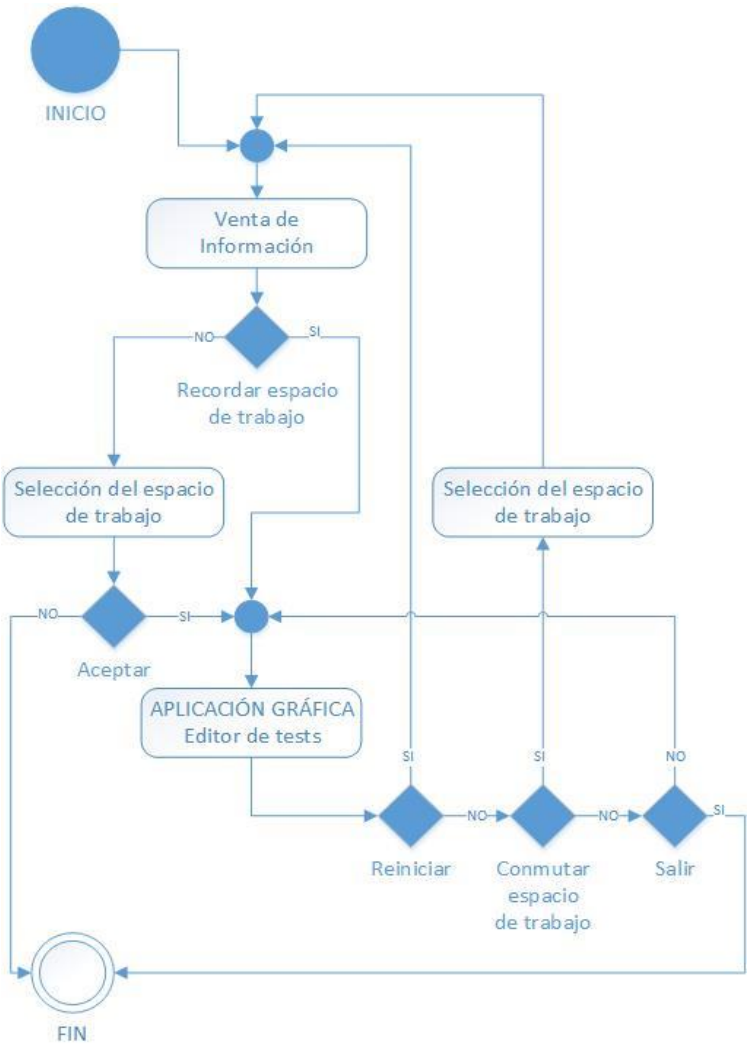


Figura 36 – Ciclo de vida de la aplicación

A continuación se desglosa el ciclo de vida en cuatro pasos.

8.1.2.1 Inicio de la aplicación

En la carpeta de la aplicación se encuentra el fichero ejecutable que inicia el entorno gráfico, denominado ***xmlGraphicTestEditor.exe***.

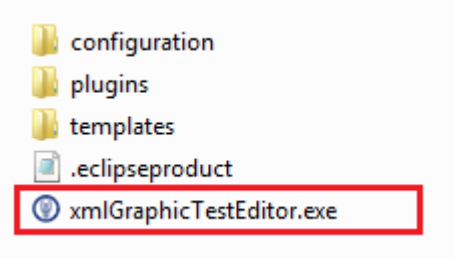


Figura 37 – Carpeta de la aplicación

Junto al fichero ejecutable se encuentra la carpeta ***templates***, dentro de la cual se pueden colocar proyectos (dentro de la carpeta ***templates/projects***) y ficheros de test (dentro de la carpeta ***templates/files***) que pueden ser utilizados como plantillas a la hora de crear estos. Estos se podrán seleccionar en de los diálogos que aparecen con las opciones de creación de proyectos de test y creación de ficheros de test dentro de la aplicación.

Además, la carpeta contiene el fichero ***.eclipseproduct*** y las carpetas ***configuración*** y ***plugins***, los cuales se recomienda no modificar ni borrar, ya que la aplicación podría dejar de funcionar.

8.1.2.2 Configuración del espacio de trabajo

Mientras se arranca la aplicación, aparecerá en la mitad de la pantalla una imagen indicando que se está inicializando el entorno.

Figura 38 – Inicialización del entorno

Una vez inicializada la aplicación, si es la primera vez que se arranca la aplicación, aparecerá el diálogo de configuración del espacio de trabajo, que no es más que la elección de una carpeta en la cual se crearán todos los proyectos de test. Se puede elegir un espacio de trabajo creado previamente, o definir uno nuevo. Además, permite elegir si este diálogo se ha de mostrar o no en futuros inicios de la aplicación.

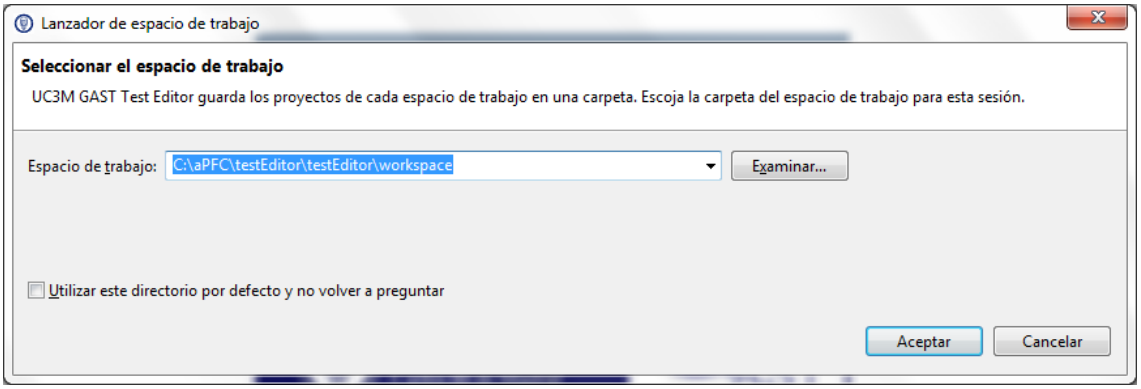


Figura 39 – Diálogo de selección del entorno de trabajo

8.1.2.3 Vista de la aplicación

Tras iniciar y configurar el espacio de trabajo, la aplicación arrancará con el aspecto por defecto o con el aspecto guardado anteriormente si no es la primera vez que se usa ese entorno de trabajo.

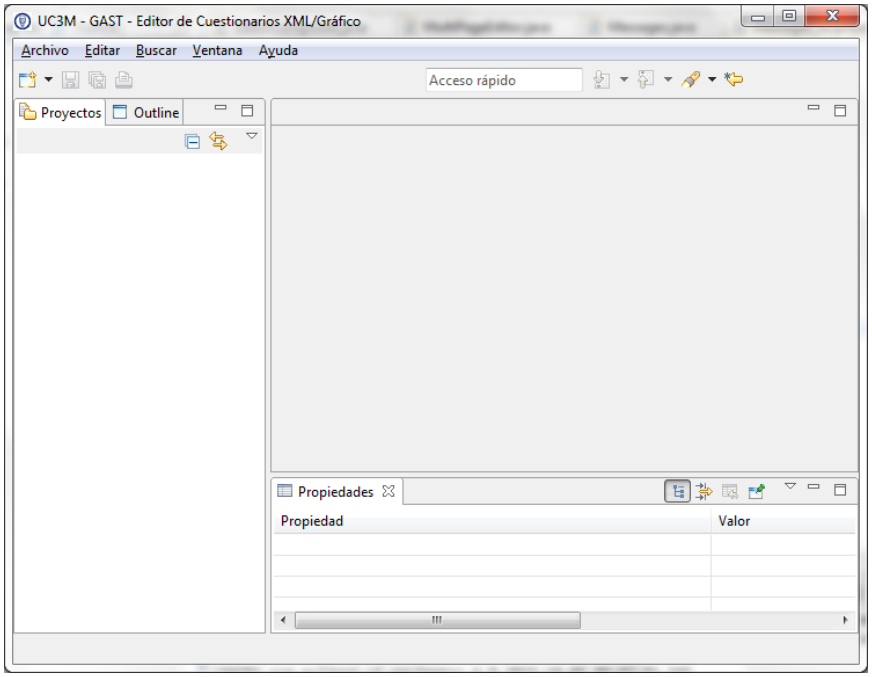


Figura 40 – Vista de la aplicación

En punto posteriores de explican cada una de las partes que conforman la aplicación, sus opciones, la creación y gestión de proyectos y ficheros de test, el editor multi-página, etc.

8.1.2.4 Reiniciar o cerrar la aplicación

El entorno gráfico dispone de opciones para reiniciar o cerrar la aplicación de diferentes formas.

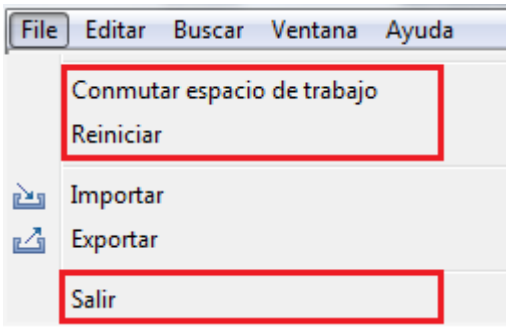


Figura 41 – Opciones de reiniciar o cerrar la aplicación

a. **Cambiar el espacio de trabajo configurado:** mediante la opción '*Archivo > Conmutar espacio de trabajo*'. Se muestra el diálogo de selección de espacio de trabajo y si se acepta, se reinicia la aplicación.

b. **Reiniciar la aplicación**: mediante la opción '*Archivo > Reiniciar*'. A continuación, si la opción 'recordar el espacio de trabajo configurado' está activa, la aplicación se reiniciará sin más. En cambio, si está desactivado, se mostrará el diálogo de elección de espacio de trabajo.

c. **Cerrar la aplicación**, mediante la opción '*Archivo > Salir*' la aplicación procederá a cerrarse. Si hay algún archivo en edición que no ha sido salvado, preguntará antes de cerrarse. Además el entorno se puede cerrar mediante el aspa roja de la esquina superior derecha, como cualquier aplicación gráfica.

8.1.3 Entorno gráfico

La aplicación se divide visualmente en partes o zonas bien diferenciadas, como se observa en la siguiente figura:

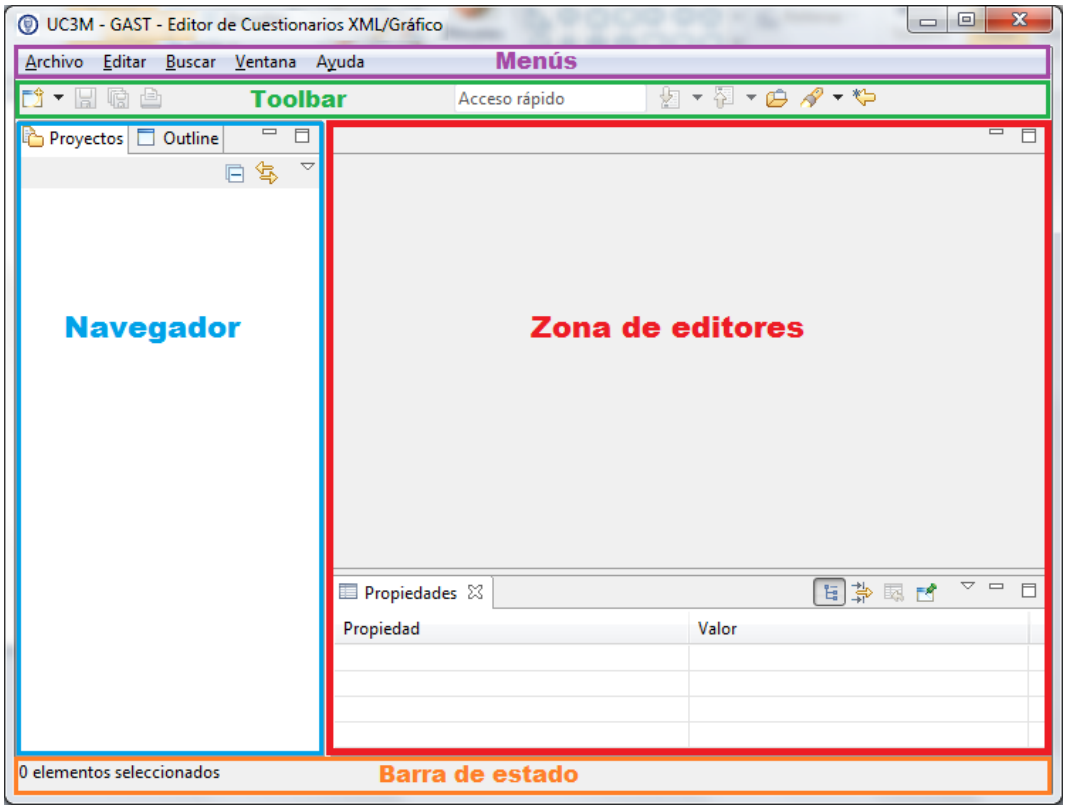


Figura 42 – Partes de la aplicación gráfica

A continuación se explican las características más importantes y las opciones con las que cuenta cada una de ellas.

8.1.3.1 Navegador

Se trata de la vista **Proyectos**, en la cual se muestran los proyectos de test, con sus carpetas, recursos gráficos (imágenes, audios, videos…) y los ficheros entre los cuales se encontrarán los ficheros de test a editar en esta aplicación. Además, es la única vista de toda la aplicación que siempre estará visible y nunca podrá ser cerrada, siempre y cuando la perspectiva por defecto esté seleccionada, aunque sí se podrá mover de ubicación dentro del entorno.

Los proyectos y todos sus recursos se muestran en forma de árbol, que puede ser desplegado o comprimido a gusto del usuario, y cuya estructura es similar a como éstos se encuentran almacenados en el espacio de trabajo configurado.

La vista dispone de un menú en su parte superior, siempre visible y con opciones que manejan la forma en cómo los proyectos se muestran y sus recursos se relacionan con las demás partes de la aplicación. Las opciones principales de este menú son aquellas que tienen que ver con la configuración de conjuntos de trabajo, que no es más que agrupar proyectos en una misma carpeta, en este caso en carpetas que agrupen proyectos de test.

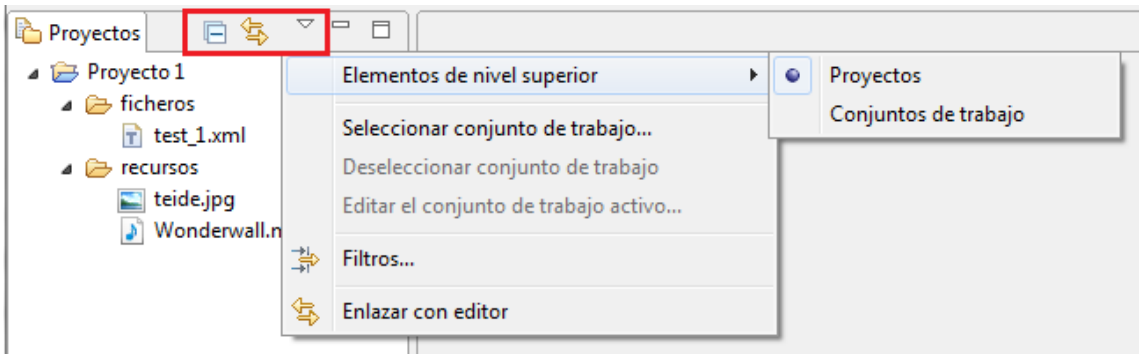


Figura 43 – Menú superior de la vista Proyectos

Contraer todo: contraerá todo el árbol al nivel configurado como superior, ya sean los proyectos o los conjuntos de trabajo.

Enlazar con editor: habilita o no el enlace del fichero que está siendo editado con su ubicación en el árbol.

Ver menú: despliega un menú con opciones de configuración:

- **Elementos de nivel superior:** permite seleccionar qué tipo de elemento es el que se muestra en el primer nivel del árbol:
  - **Proyectos**
  - **Conjuntos de trabajo**
- **Seleccionar conjunto de trabajo**: muestra el diálogo que permite seleccionar los conjuntos de trabajo que se muestran y cuales no en el árbol de la vista. Además, permite crear, editar y borrar conjuntos de trabajo. Ver sección 8.1.2.2.
- **Deseleccionar conjunto de trabajo**: hace que el conjunto de trabajo seleccionado en el árbol se oculte y no aparezca.
- **Editar el conjunto de trabajo activo**: muestra el diálogo de edición del conjunto de trabajo seleccionado en el árbol, permitiendo definir su nombre y qué proyectos se muestran bajo él.

Filtros: abre el diálogo que permite definir filtros a aplicar sobre el árbol y la visualización de recursos.

Enlazar con editor: habilita o no el enlace del fichero que está siendo editado con su ubicación en el árbol.

Además, la vista cuenta con un menú contextual que proporciona opciones que se aplicarán sobre los recursos seleccionados. En función del elemento seleccionado del árbol, aparecerán unas u otras opciones.

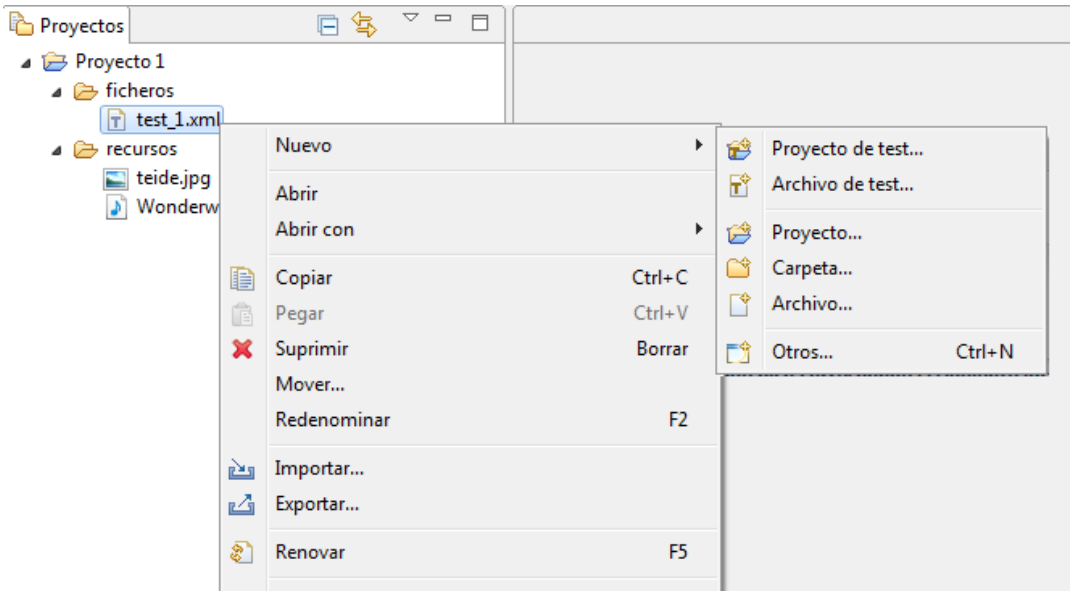


Figura 44 – Menú contextual (parcial) de la vista Proyectos

- **Nuevo:** menú con opciones para crear nuevos recursos:

  - **Proyecto de test:** muestra el diálogo de creación de un proyecto de test, el cual permite seleccionar una plantilla con una estructura predeterminada formada por carpetas, ficheros de test e incluso recursos gráficos. El diálogo completo se explica en la sección 8.1.4.1.

  - **Archivo de test**: muestra el diálogo de creación de un archivo de test, el cual también permite seleccionar una plantilla por defecto. El diálogo completo se explica en la sección 8.1.4.2.

  - **Proyecto**: abre el diálogo común de creación de un proyecto.

  - **Carpeta**: abre el diálogo común de creación de una carpeta.

  - **Archivo**: abre el diálogo común de creación de un archivo.

  - **Otros**: muestra el diálogo (Figura 56) en el cual seleccionar el tipo de recurso a crear de entre todos los posibles, donde se encuentran los anteriores, entre otros.

- **Abrir**: abrir el fichero seleccionado con el editor por defecto para ese tipo de archivo.

- **Abrir con**: permite seleccionar de una lista de editor con cuál de ellos se abre el archivo.

 **Copiar**: copiar los elementos seleccionados.

 **Pegar**: pegar los elementos en la ubicación seleccionada.

 **Suprimir**: eliminar los elementos seleccionados.

- **Mover**: abre un diálogo que permite seleccionar la nueva ubicación del elemento seleccionado.

- **Renombrar**: muestra un diálogo que permite renombrar el elemento seleccionado.

 **Importar**: muestra el diálogo que permite elegir el tipo de recurso que se quiere importar. Más información en la sección 8.1.4.5.

 **Exportar**: abre el diálogo que permite elegir el tipo de recurso que se desea exportar. Más información en la sección 8.1.4.4.

 **Renovar**: opción que refresco del contenido de la vista.

- **Abrir/Cerrar el proyecto**: en función del estado del proyecto (abierto o cerrado), mostrará la acción contraria.

Además, existen otras opciones que tienen que ver con la configuración de un repositorio de contenidos que permita el control de versiones. Sobre ellas se muestra más información en la sección 8.1.4.6, y son:

- **Trabajo en equipo:** menú que dispone de numerosas opciones, que varían en función de si el elemento seleccionado forma ya o no parte de un repositorio de contenidos.

- **Comparar con:** permite comparar el elemento seleccionado con la versión almacenada en el repositorio de contenidos, con versiones locales, etc.

- **Sustituir por:** da la opción de cambiar el contenido del elemento seleccionado por otra versión, ya sea local o almacenada en el repositorio.

A estas opciones se unen otras menos importantes. La vista además permite el uso de ciertas teclas para realizar algunas opciones, como pueden ser las de copiar, pegar, eliminar o renombrar elementos, refrescar la vista, etc.

También está permitido arrastrar y soltar elementos del árbol de unas ubicaciones a otras, si bien se recomienda tener precaución con este tipo de opciones, ya que se puede llegar a modificar erróneamente la estructura de los proyectos o ciertos ficheros de test pueden dejar de funcionar correctamente si estos contienen rutas a elementos del proyecto modificado.

8.1.3.2 Menú superior

Se trata de las opciones disponibles en la parte superior de la aplicación. Este menú está compuesto a su vez de diferentes sub menús, los cuales contienen una serie de opciones separadas según su finalidad.

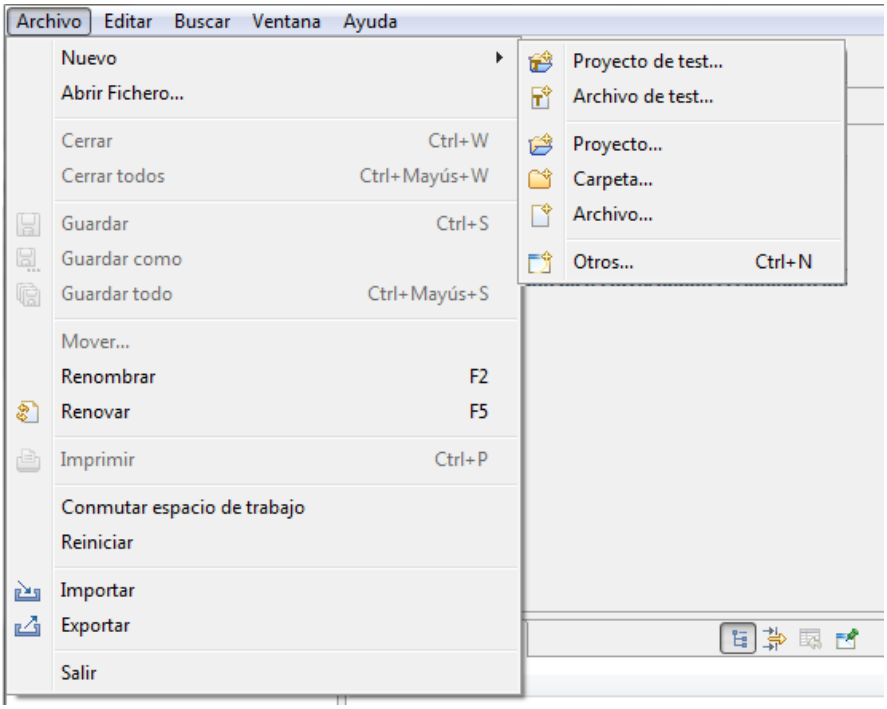


Figura 45 – Menú superior

Archivo

Se trata del menú con las opciones principales:

- **Nuevo:** menú con opciones para crear nuevos recursos:

  Proyecto de test: muestra el diálogo de creación de un proyecto de test, el cual permite seleccionar una plantilla con una estructura predeterminada formada por carpetas, ficheros de test e incluso recursos gráficos. El diálogo completo se explica en la sección 8.1.4.1.

  Archivo de test: muestra el diálogo de creación de un archivo de test, el cual también permite seleccionar una plantilla por defecto. El diálogo completo se explica en la sección 8.1.4.2.

  Proyecto: abre el diálogo común de creación de un proyecto.

  Carpeta: abre el diálogo común de creación de una carpeta.

Archivo: abre el diálogo común de creación de un archivo.

Otros: muestra el diálogo (Figura 56) en el cual seleccionar el tipo de recurso a crear de entre todos los posibles, donde se encuentran los anteriores, entre otros.

- **Abrir fichero:** muestra el diálogo por defecto de búsqueda de un fichero dentro del equipo, el cual se abrirá en la aplicación con el editor o la aplicación por defecto asociada a su formato.
- **Cerrar:** cierra el editor seleccionado actualmente.
- **Cerrar todos:** cierra todos los editores abiertos.

Guardar: salvar el fichero que se está editando en ese momento. Estará activo si el fichero editado actualmente ha sido modificado.

Guardar como: abrir el diálogo de almacenamiento, en el cual elegir el formato de almacenamiento del fichero.

Guardar todo: salvar todos los ficheros que están siendo editados. Opción habilitada si al menos uno de los ficheros editados ha sido modificado.

- **Mover:** abre un diálogo que permite seleccionar la nueva ubicación del elemento seleccionado.
- **Renombrar**: muestra un diálogo que permite renombrar el elemento seleccionado.

Renovar: opción que refresco del contenido de la vista seleccionada.

Imprimir: muestra el diálogo de impresión que permite imprimir el fichero que se está editando. Habilitada si el editor seleccionado así lo requiere.

- **Conmutar espacio de trabajo:** abre el diálogo (Figura 39) de selección del espacio de trabajo sobre el cual se almacenarán los proyectos de test creados.
- **Reiniciar:** reinicia el entorno gráfico. Si existe algún fichero que esté siendo editado, preguntará antes si se quiere o no guardar los cambios.

Importar: muestra el diálogo que permite elegir el tipo de recurso que se quiere importar. Más información en la sección 8.1.4.5.

 **Exportar**: abre el diálogo que permite elegir el tipo de recurso que se desea exportar. Ver más en la sección 8.1.4.4.

- **Salir:** cierra la aplicación. Si existe algún fichero que esté siendo editado, preguntará si se desea o no guardar los cambios.

Editar

Menú con las opciones de edición:

 **Deshacer**: deshace los cambios realizados.

 **Rehacer:** rehace las modificaciones efectuadas.

 **Cortar:** copia al portapapeles y elimina el elemento o el texto seleccionado.

 **Copiar**: copia al portapapeles el elemento o el texto seleccionado.

 **Pegar**: pega los elementos o el texto copiado en el portapapeles.

 **Borrar**: elimina los elementos o el texto seleccionado, mostrando antes un diálogo de confirmación.

- **Seleccionar todo:** seleccionar todos los elementos de la vista activa, o todo el texto si se trata de un editor.
- **Buscar y reemplazar:** muestra un diálogo que permite buscar un texto en un editor y reemplazarlo por otro.

Buscar

Menú con opciones relativas a la búsqueda de texto en el entorno:

- **Abrir diálogo Buscar:** muestra el diálogo (Figura 46) avanzado de búsqueda, que permite realizar búsquedas en todo el espacio de trabajo o únicamente en una parte de este. Además, permite reemplazar un texto busca por otro en todo el espacio de trabajo.
- **Texto:** menú con opciones que permiten buscar un texto en diferentes ubicaciones:
  - **Buscar texto en el espacio de trabajo:** en todos los proyectos.
  - **Buscar texto en el proyecto:** en el proyecto seleccionado.
  - **Buscar texto en el archivo:** en el archivo editado.

- **Buscar texto en el conjunto de trabajo:** dentro de la agrupación de proyectos seleccionada. Más información en la sección 8.1.4.7.

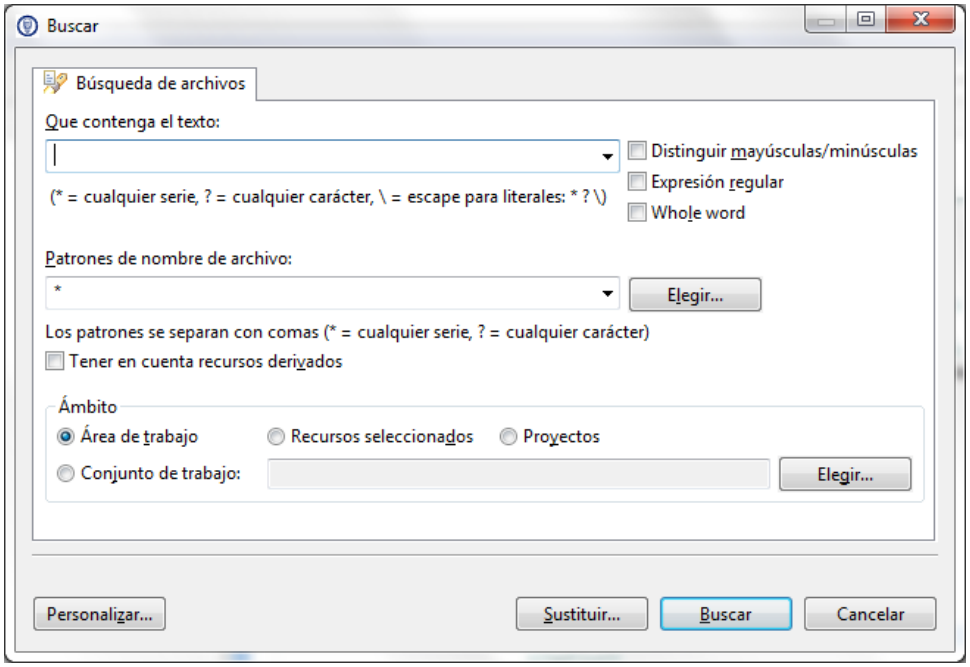


Figura 46 – Diálogo de búsqueda avanzada

Ventana

Menú con opciones relativas a la presentación de la aplicación, sus perspectivas y las vistas:

- **Ocultar barra de herramientas:** opción que muestra u oculta la Toolbar (sección 8.1.3.3) del entorno gráfico.
- **Mostrar vista:** abre el diálogo (Figura 47) que muestra la lista de vistas disponibles, separadas por categorías.
- **Mostrar perspectiva:** abre el diálogo (Figura 48) con la lista de perspectivas definidas.
- **Restablecer perspectiva:** restablece la perspectiva seleccionada a su apariencia por defecto.

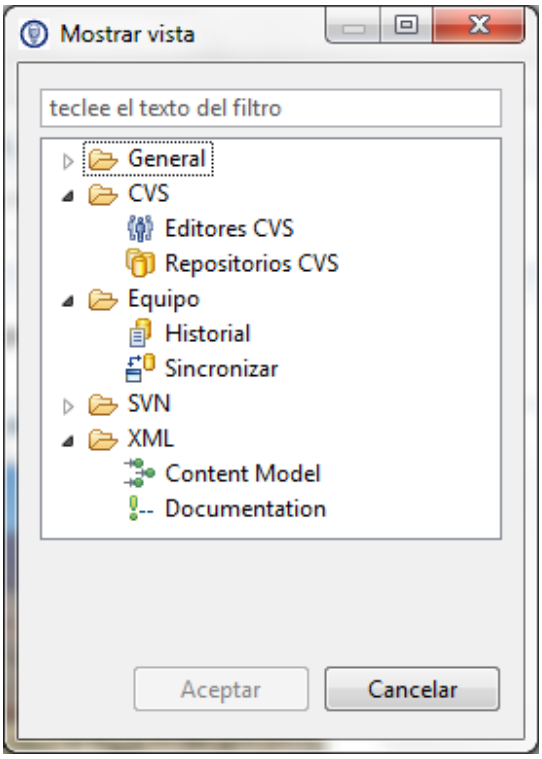


Figura 47 - Vistas

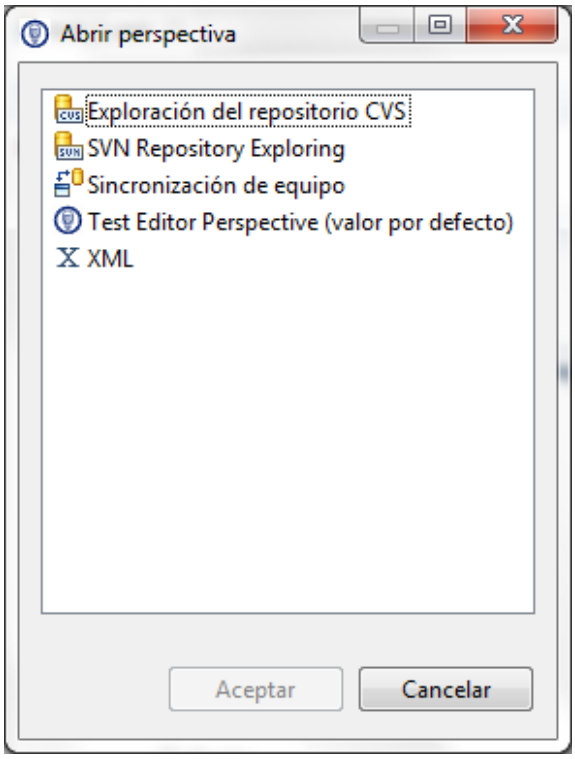


Figura 48 - Perspectivas

Ayuda

Menú que proporciona información acerca de la aplicación:

Ayuda: abre una ventana a parte de la aplicación, formada por un árbol a la izquierda con el árbol de apartados definidos en este manual, y a la derecha la visualización del apartado seleccionado.

Buscar ayuda: abre la vista de **Ayuda** dentro de la aplicación. Ésta permite introducir un texto a buscar dentro de este manual.

- **Ayuda dinámica:** abre la vista de **Ayuda** dentro de la aplicación.
- **Acerca de**: abre un diálogo con información acerca de la versión de la aplicación y los plugins que la componen.

8.1.3.3 Toolbar

Barra de herramientas situada bajo la barra de menús de la aplicación. Contiene enlaces cortos a las opciones más utilizadas de la aplicación, además de herramientas utilizadas en el editor de ficheros.

Figura 49 – Toolbar de la aplicación

Nuevo: muestra el diálogo (Figura 56) en el cual seleccionar el tipo de recurso a crear de entre todos los posibles, incluidos los siguientes. Además muestra un menú con opciones para crear nuevos recursos del tipo:

- **Proyecto de test:** muestra el diálogo de creación de un proyecto de test, el cual permite seleccionar una plantilla con una estructura predeterminada formada por carpetas, ficheros de test e incluso recursos gráficos. El diálogo completo se explica en la sección 8.1.4.1.
- **Archivo de test**: muestra el diálogo de creación de un archivo de test, el cual también permite seleccionar una plantilla por defecto. El diálogo completo se explica en la sección 8.1.4.2.
- **Proyecto**: abre el diálogo común de creación de un proyecto.
- **Carpeta**: abre el diálogo común de creación de una carpeta.
- **Archivo**: abre el diálogo común de creación de un archivo.
- **Otros**: muestra el diálogo (Figura 56) en el cual seleccionar el tipo de recurso a crear de entre todos los posibles, donde se encuentran los anteriores, entre otros.

Guardar: salvar el fichero que se está editando en ese momento. Estará activo si el fichero editado actualmente ha sido modificado.

Guardar todo: salvar todos los ficheros que están siendo editados. Opción habilitada si al menos uno de los ficheros editados ha sido modificado.

Imprimir: muestra el diálogo de impresión que permite imprimir el fichero que se está editando. Habilitada si el editor seleccionado así lo requiere.

En cuanto a la barra de herramientas del editor, las opciones sirven para realizar búsquedas dentro del archivo que está siendo editado, moverse por este, etc.

8.1.3.4 Barra de estado

Se trata de la barra inferior de la aplicación, en la cual se muestra dinámicamente información acerca de los elementos seleccionados en las demás vistas.

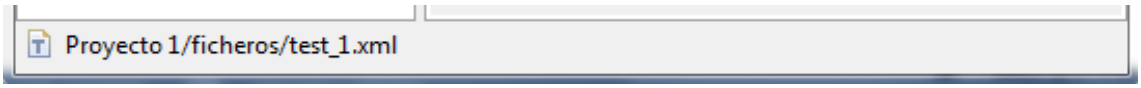


Figura 50 – Barra de estado de la aplicación

8.1.3.5 Zona de edición

Denominada así porque es el espacio reservado dentro del cual se irán visualizando los ficheros de test que se vayan editando. Además es posible colocar vistas en esa zona, como por defecto lo hace la vista de **Propiedades.**

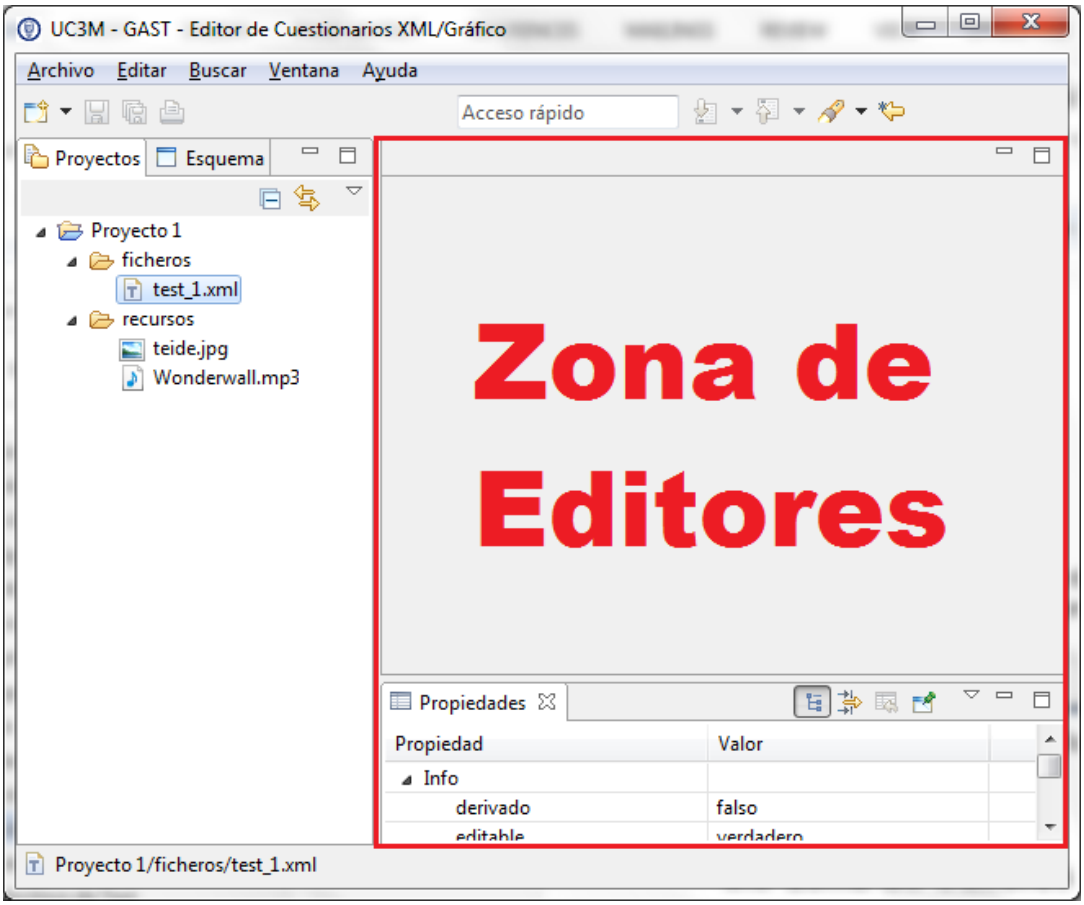


Figura 51 – Espacio reservado a editores

8.1.4 Proyectos y Archivos de Test

Existen diferentes formas de crear, añadir o gestionar los proyectos de test o los propios ficheros de test incluidos en el espacio de trabajo configurado.

8.1.4.1 Creación de un proyecto de test

Se puede crear un proyecto de test a partir de una plantilla seleccionada en el diálogo abierto desde la opción ***Nuevo > Proyecto de test*** disponible en el menú *Archivo*, en la *toolbar* o en el menú contextual de la vista de *Proyectos*.

Dicho diálogo consta de dos páginas:

- La primera de esas páginas, denominada **Proyecto**, solicita obligatoriamente un nombre de proyecto, y da la posibilidad de elegir una alternativa al espacio de trabajo configurado para almacenar el proyecto. Si el nombre del proyecto o la ubicación introducida no es correcto, se mostrará un mensaje alertando de ello y desactivando los botones inferiores del diálogo. Si todo es correcto, existen tres posibilidades:
    - *Siguiente*: se pasa a la segunda página del diálogo.
    - *Finalizar*: se crea el proyecto con los parámetros introducidos y con la plantilla seleccionada en la segunda página, siendo la de por defecto si no se ha llegado a seleccionar ninguna.
    - *Cancelar*: se cierra el diálogo y no se crea ningún proyecto.

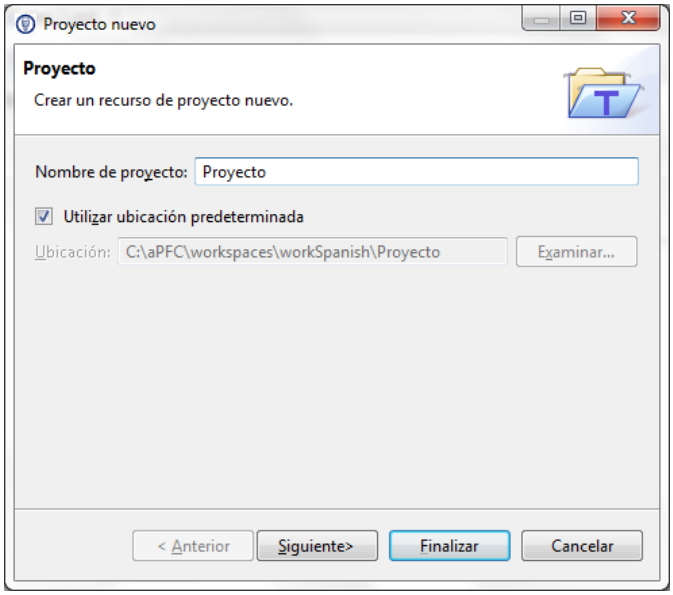


Figura 52 – Proyecto nuevo de test (primera página)

- La segunda página, denominada **Plantilla**, permite seleccionar de una lista la estructura de fichero que se desea añadir al nuevo proyecto. Dicha estructura está compuesta de carpetas, ficheros de test y alguna de ellas, incluso por recursos gráficos como imágenes, audios o videos, y ésta se puede pre-visualizar al lado de la lista de plantillas. Esta lista contiene por defecto una serie de proyectos incrustados en la propia aplicación, pero el usuario tiene la posibilidad de añadir sus propias plantillas en la carpeta *templates/projects* localizada en la carpeta de la distribución, junto al ejecutable de la aplicación. Una vez seleccionada la plantilla, se podrá volver a la primera página del diálogo, finalizar o cancelar la creación del proyecto.

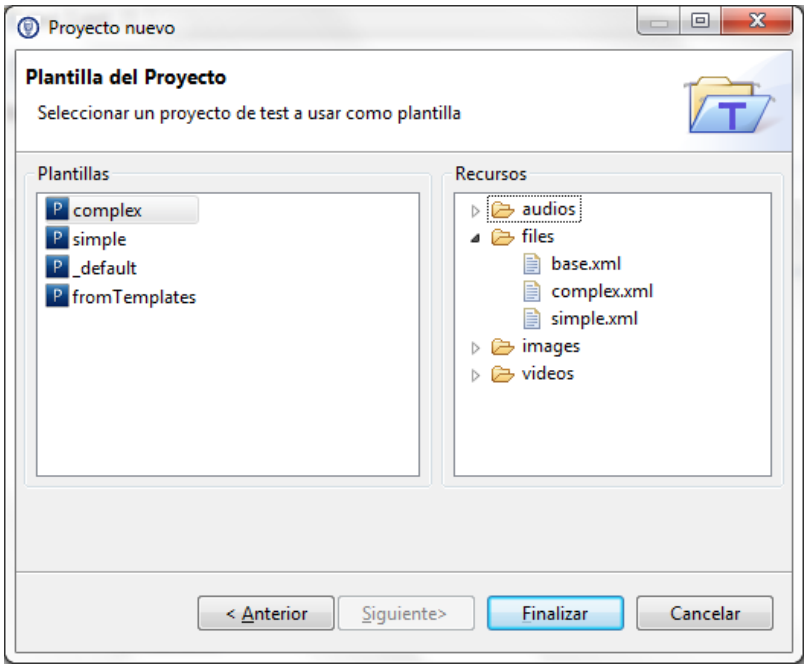


Figura 53 - Proyecto nuevo de test (segunda página)

8.1.4.2 Creación de un archivo de test

Se puede crear un archivo de test a partir de una plantilla seleccionada en el diálogo abierto desde la opción ***Nuevo > Archivo de test*** disponible en el menú *Archivo*, en la *toolbar* o en el menú contextual de la vista de *Proyectos*.

Dicho diálogo consta de dos páginas:

- La primera de esas páginas, denominada **Archivo**, solicita obligatoriamente un nombre de fichero y una ubicación para este, la cual se puede seleccionar del árbol con la estructura de proyectos mostrado en el diálogo. Si el nombre del fichero o la ubicación introducida no es correcta, se mostrará un mensaje alertando de ello y desactivando los botones inferiores

del diálogo. Si todo es correcto, se podrá pasar a la segunda página del diálogo mediante la opción *Siguiente* o bien *Cancelar*, acción que se cierra el diálogo y no crea ningún archivo.

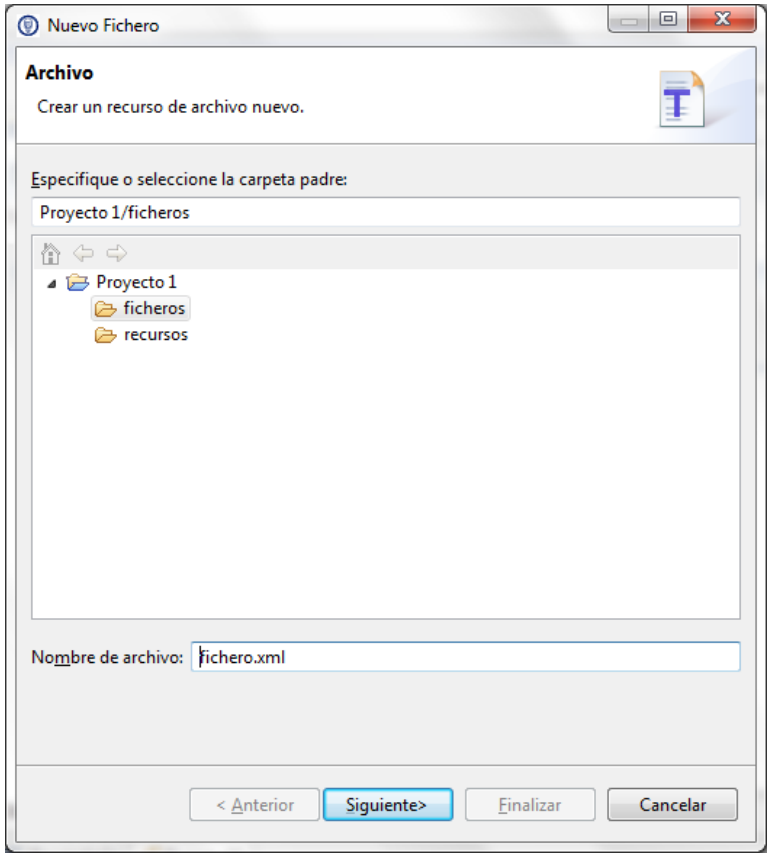


Figura 54 – Archivo nuevo de test (primera página)

- La segunda página, denominada **Plantilla**, solicita obligatoriamente un título para el test y permite introducir una descripción de este. Además, da la posibilidad de seleccionar un contenido por defecto a como de plantilla de entre la lista de ficheros mostrados. Esta lista contiene por defecto una serie de proyectos incrustados en la propia aplicación, pero el usuario tiene la posibilidad de añadir sus propias plantillas en la carpeta *templates/files* localizada en la carpeta de la distribución, junto al ejecutable de la aplicación. Una vez completado el diálogo y seleccionada una plantilla, se podrá:

    - *Anterior*: se vuelve a la primera página del diálogo.
    - *Finalizar*: se crea el archivo de test con los parámetros introducidos y la plantilla por defecto como contenido.
    - *Cancelar*: se cierra el diálogo y no se crea ningún archivo.

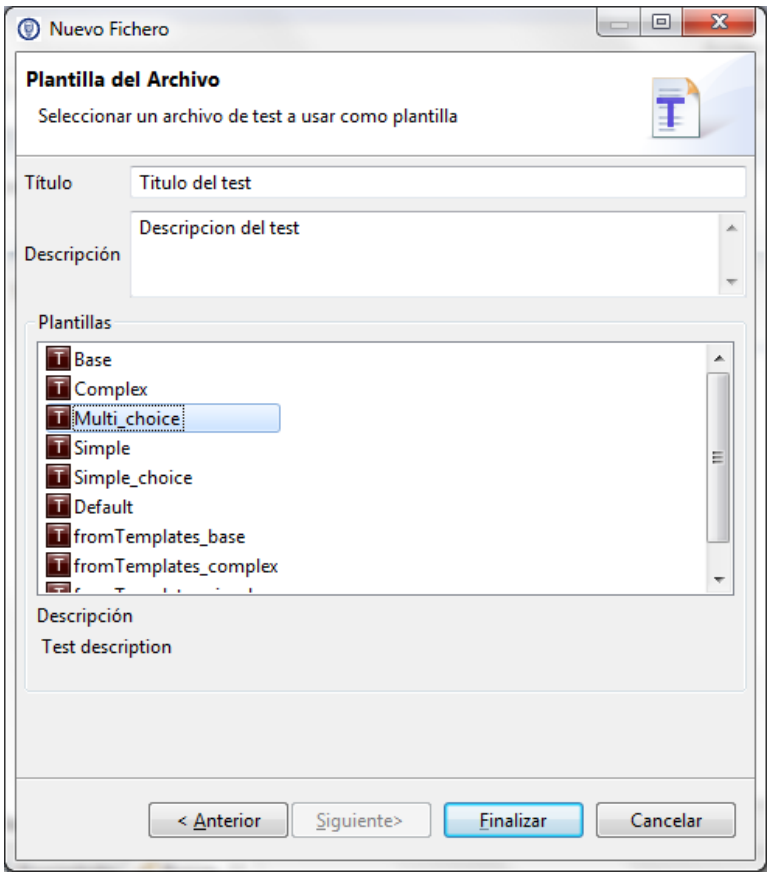


Figura 55 - Archivo nuevo de test (segunda página)

8.1.4.3 Creación de otros recursos

Además de los diálogos de creación de un proyecto y un fichero de test a través de plantillas, es posible crear un proyecto o un fichero usando el diálogo de creación por defecto de cada uno de ellos que proporciona Eclipse.

Esta opción, junto con la de crear un directorio nuevo, están disponibles desde la opción ***Nuevo > [opción]*** disponible en el menú *Archivo*, en la *toolbar* o en el menú contextual de la vista de *Proyectos*.

Los diálogos son similares a los anteriormente descritos, únicamente con la primera de las páginas.

Además, se da la posibilidad de crear cualquier tipo de recurso de los anterior, incluso otros muchos tipos, mediante la opción ***Nuevo >Otros*** disponible en el menú *Archivo*, en la *toolbar* o en el menú contextual de la vista de *Proyectos*. Esta mostrará un diálogo en el cual se puede seleccionar el tipo de recurso a crear de una larga lista, mostrando el diálogo de creación configurado para cada tipo.

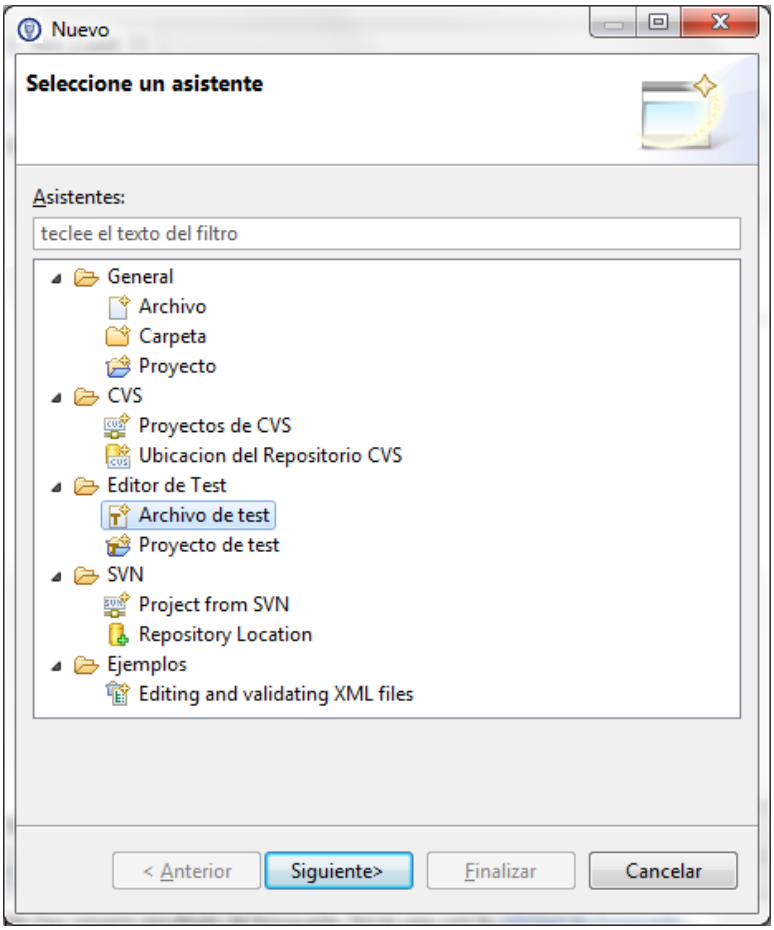


Figura 56 – Diálogo de creación de cualquier tipo de recurso

8.1.4.4 Exportar

Es posible exportar cualquier tipo de recurso seleccionado en el árbol de la vista de *Proyectos*. Para ello se ha de seleccionar la opción ***Exportar*** disponible en el menú *Archivo* o en el menú contextual de la propia vista de *Proyectos*.

La primera página del diálogo da la opción de elegir la forma en cómo se va a exportar el recurso seleccionado. Se recomienda seleccionar la opción *Sistema de archivos,* aunque también podría ser útil la de *Archivo de archivados*, que generará un archivo comprimido con el recurso o recursos seleccionados.

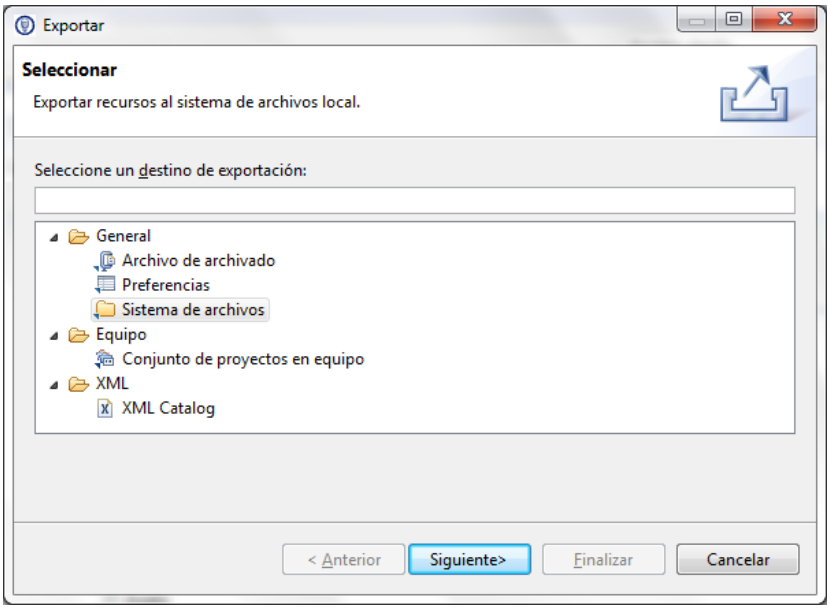


Figura 57 – Diálogo de exportar un recurso (primera página)

La segunda página del diálogo *Exportar* permite seleccionar los recursos que se van a exportar, así como el directorio en el cual se depositarán. También dispone de opciones para elegir si se sobrescriben o no los archivos existentes sin avisar, en caso de coincidencia, o si se ha de crear la estructura del proyecto hasta el fichero seleccionado o únicamente se copian los ficheros seleccionados.

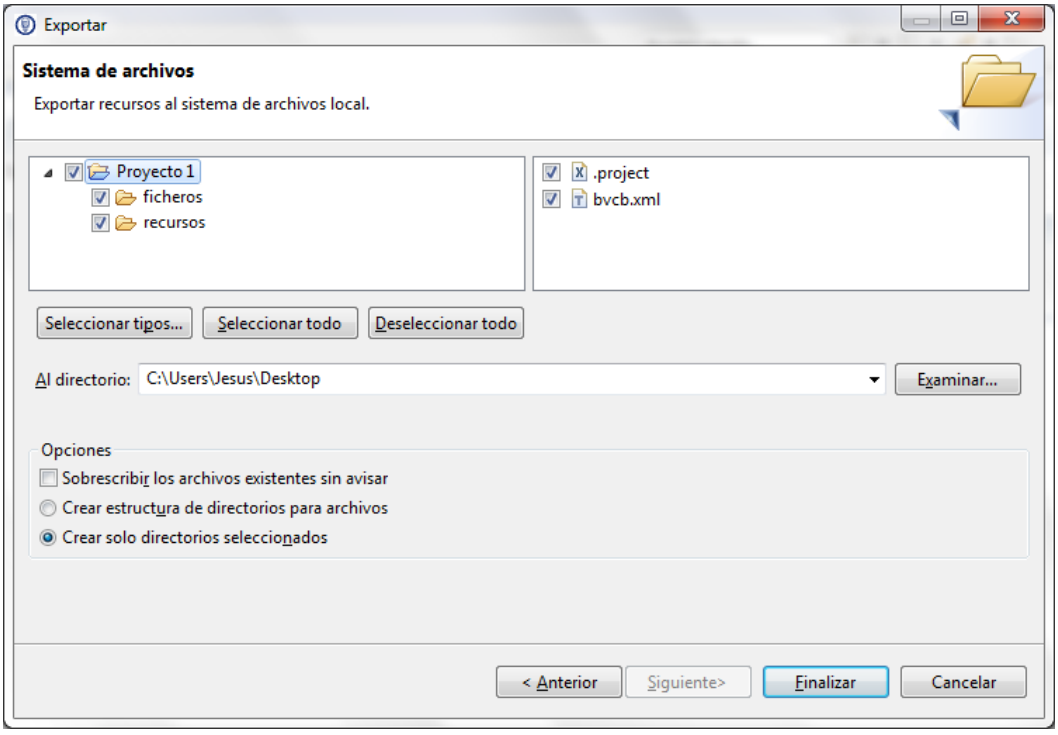


Figura 58 - Diálogo de exportar un recurso (segunda página)

8.1.4.5 Importar

Es posible importar cualquier tipo de recurso seleccionado en el sistema de ficheros del equipo al árbol de la vista de *Proyectos*, y de este modo al espacio de trabajo. Para ello se ha de seleccionar la opción ***Importar*** disponible en el menú *Archivo* o en el menú contextual de la propia vista de *Proyectos*.

La primera página del diálogo da la opción de elegir desde dónde se va a importar el recurso seleccionado. Las recomendadas son las que dicen *Proyectos existentes en el espacio de trabajo* o *Sistema de Archivos*, aunque también es válida la de *Archivo de archivado*, que lo hará a partir de un archivo comprimido.

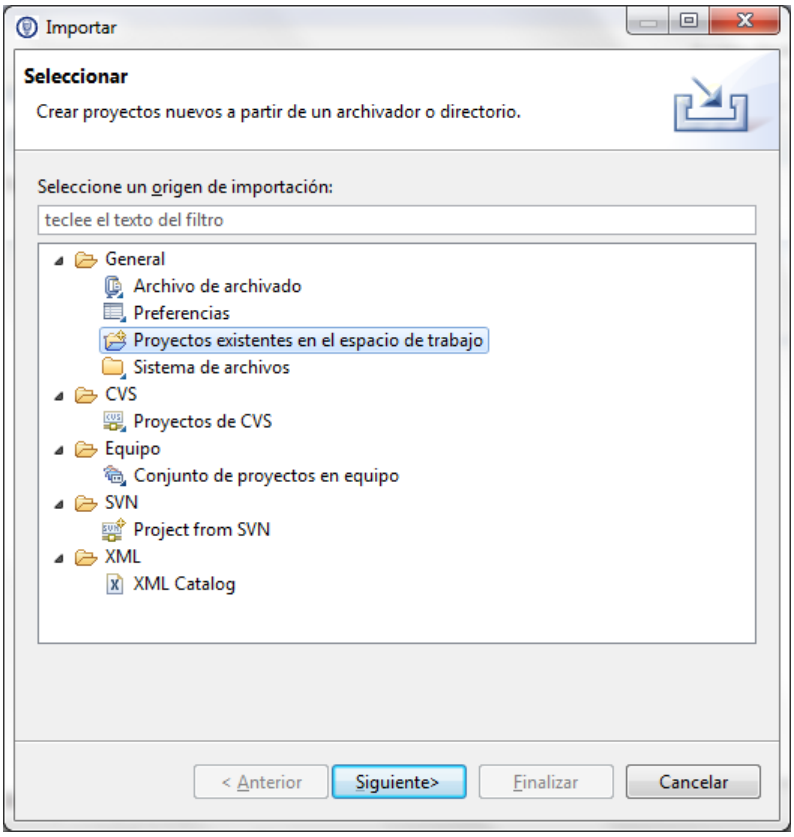


Figura 59 – Diálogo de importar un recurso (primera página)

Además este diálogo da la oportunidad de importar un recurso desde un repositorio de contenidos CVS o SVN, que habrá que configurar previamente. En el siguiente punto se habla de cómo proceder para configurar y trabajar con este tipo de repositorio.

Si la opción elegida fue *Proyectos existentes en el espacio de trabajo*, decir que únicamente permite importar proyectos, no ficheros.

A continuación se muestra el diálogo con las diferentes opciones, en el que es posible seleccionar una ruta a un espacio de trabajo, y de este seleccionar que proyectos se importan y cuáles no. También se podrá hacer lo propio desde una ruta a un archivo comprimido que contenga proyectos en su interior.

Las opciones permiten definir si se copia o no el proyecto o simplemente se referencia a su ubicación original, algo peligroso en caso de ser borrado, y también si se incluyen a un conjunto de trabajo definido en el navegador.

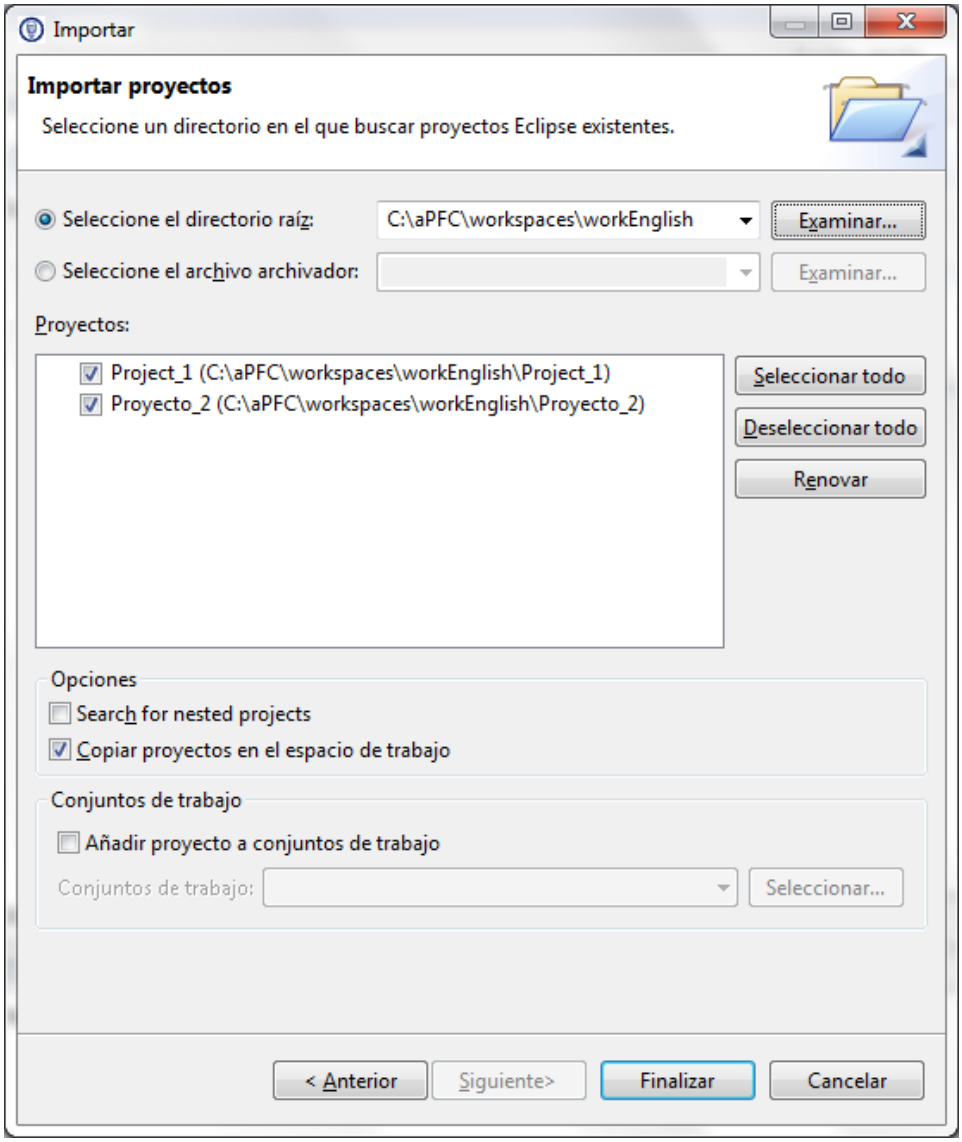


Figura 60 – Diálogo de importar un proyecto desde un espacio de trabajo

8.1.4.6 Compartir

A la hora de compartir proyectos se han configurado dos tipos posibles de repositorios de contenido: CVS y SVN. Ambos son muy similares en cuanto a las opciones que aportan a la aplicación.

Para ello se ha de seleccionar la opción ***Trabajo en Equipo > Compartir proyecto*** disponible en el menú contextual de la propia vista de *Proyectos*.

La primera página del diálogo da la opción de elegir qué tipo de repositorio de los disponibles se desea usar para compartir el proyecto: CVS o SVN.

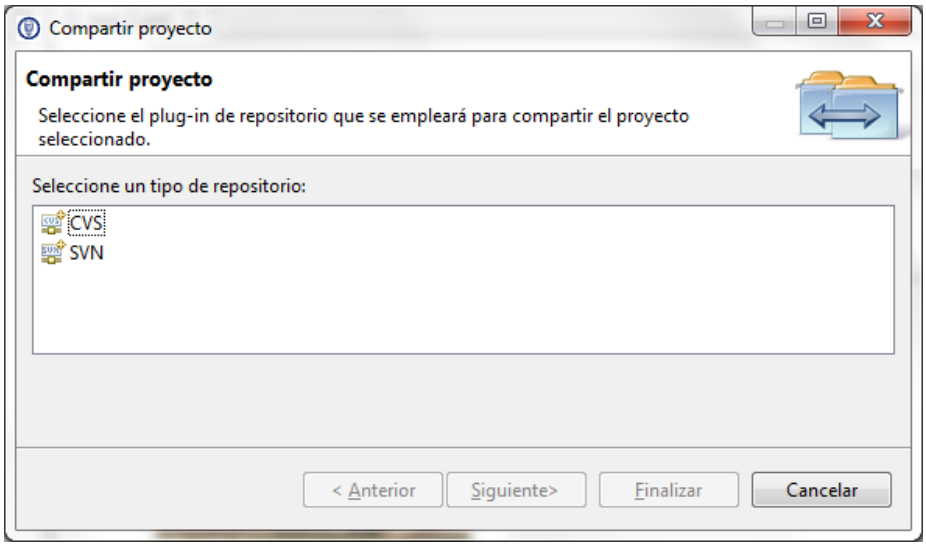


Figura 61 – Diálogo de compartir proyecto (primera página)

Si la opción elegida ha sido CVS, si no se ha configurado ningún repositorio de deberá proceder a ello.

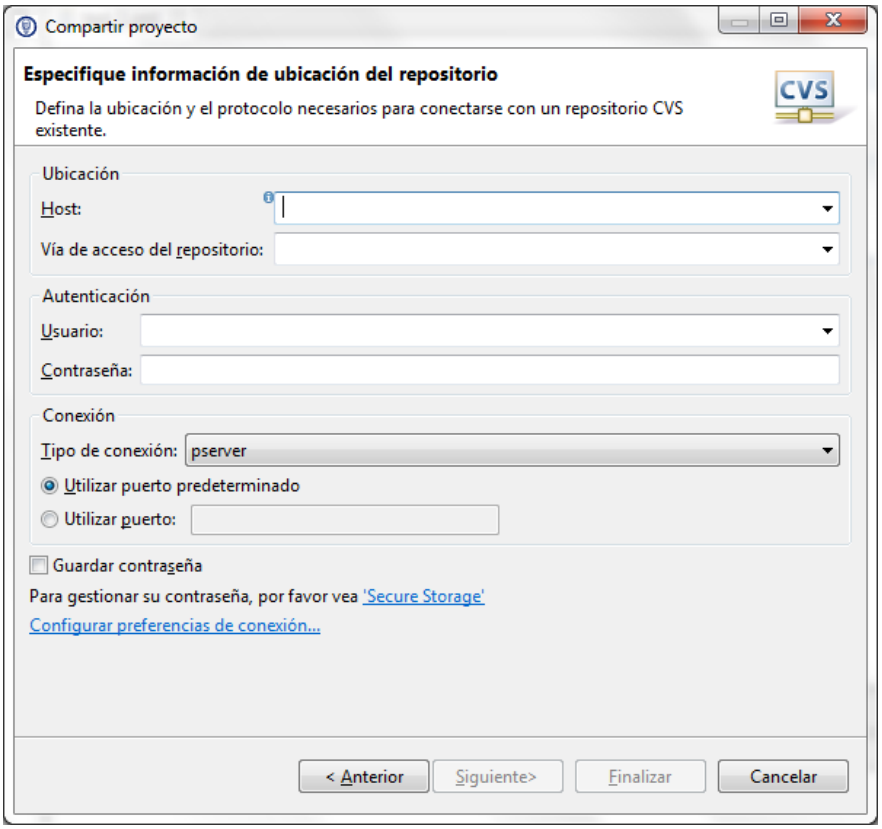


Figura 62 – Diálogo configuración CVS

Si la opción elegida fue SVN y tampoco se ha configurado ningún repositorio, se debe realizar primero.

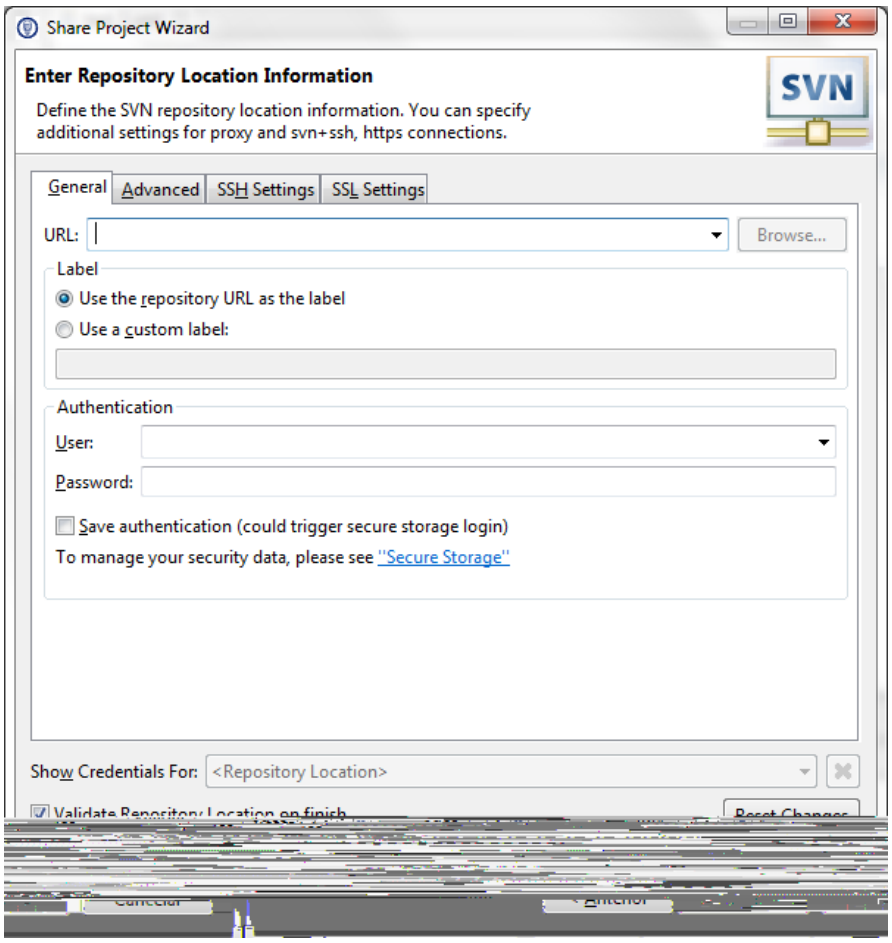


Figura 63 – Diálogo de configuración de SVN

Un vez configurados el repositorio correspondiente se procederá a elegir la ubicación exacta sobre la cual compartir el proyecto y de esta manera crear la primera de las revisiones del proyecto.

Posteriormente se podrán ir creando más versiones e ir revisando los cambios, volver a versiones anteriores, crear copias de seguridad, etc.

8.1.4.7 Gestión

Los proyectos de test se pueden gestionar en la vista de *Proyectos*, de manera que es posible cerrarlos, visualizar sus propiedades, renombrarlos, etc. Todas esas opciones se encuentran en el menú contextual de la vista.

Además, se puede cambiar la forma en cómo estos se visualizan en el árbol de la vista, de manera que pueden ser agrupados en carpetas llamadas *Conjuntos de trabajo*. Estos conjuntos se pueden crear, modificar, gestionar o eliminar desde las opciones que se encuentran en el menú superior de la vista, de las cuales se habla en el apartado 3.1 de este manual.

El diálogo de configuración de los conjuntos de trabajo en el siguiente:

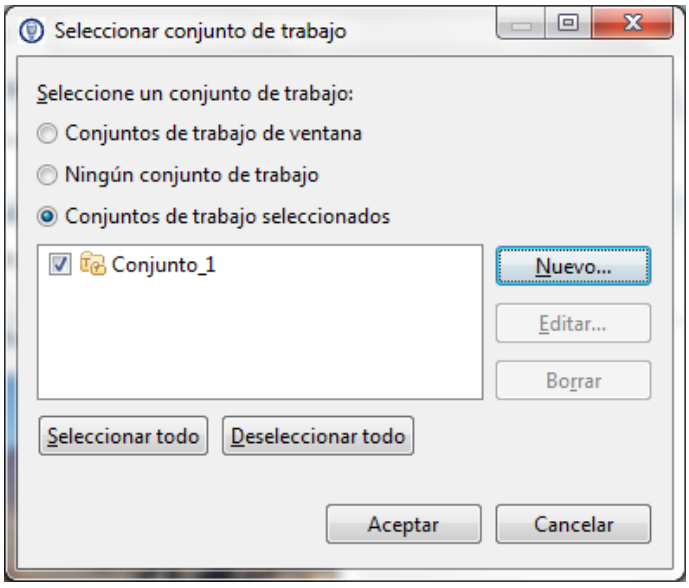


Figura 64 – Diálogo de selección de un conjunto de trabajo

Se podrán crear nuevos conjuntos de trabajo mediante la opción ***Nuevo*** de este diálogo, ***Editar*** uno ya creado o ***Borrar*** el seleccionado.

En el caso de que se desee crear un nuevo conjunto de trabajo, se ha de seleccionar la opción *Proyectos de Test* que aparece en la lista.

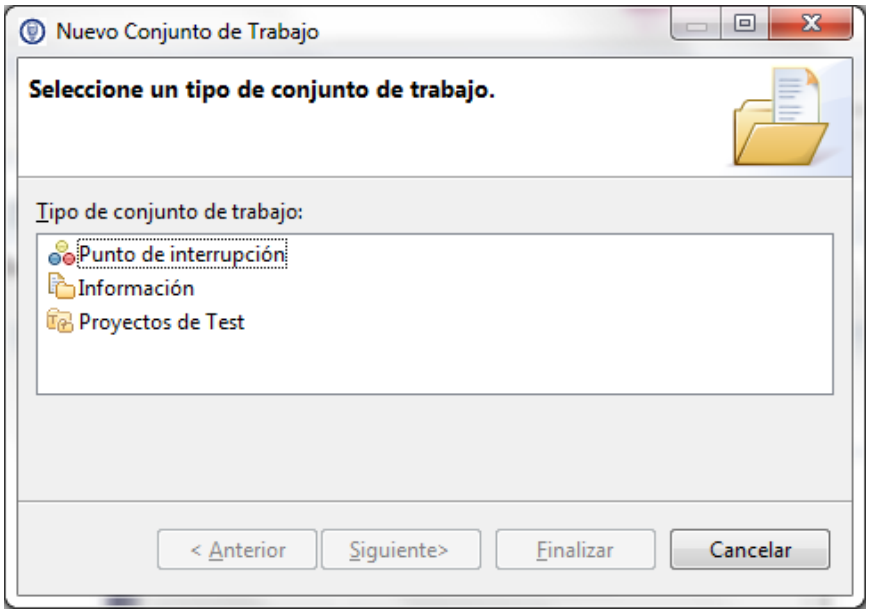


Figura 65 – Diálogo de nuevo conjunto de trabajo (primera página)

Una vez seleccionado el tipo de conjunto de trabajo a crear, en la segunda página se podrán seleccionar aquellos proyectos que se desea formen parte del conjunto de trabajo.

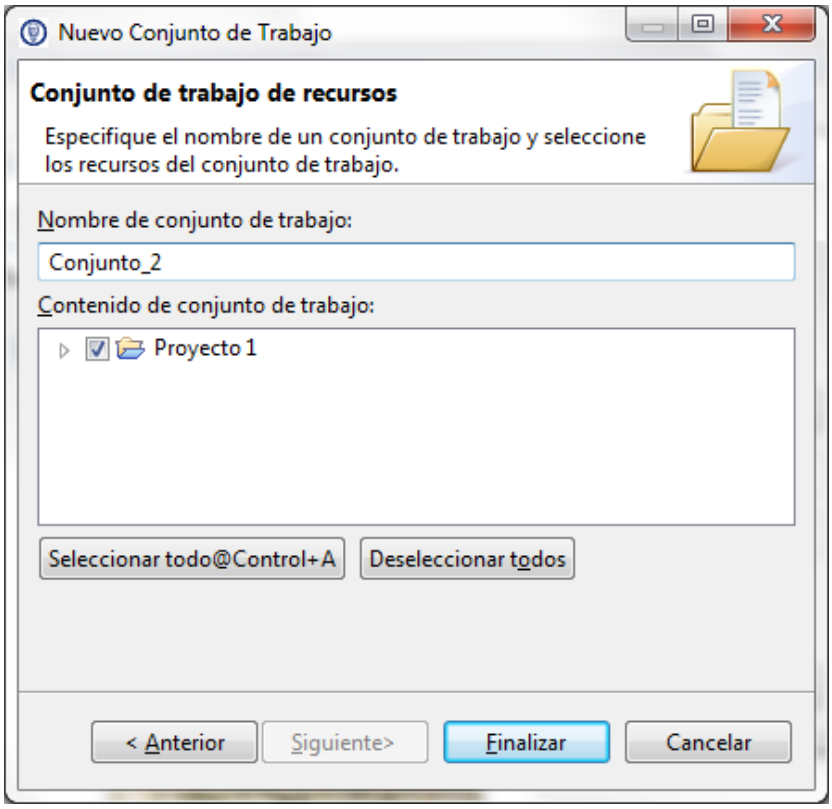


Figura 66 - Diálogo de nuevo conjunto de trabajo (segunda página)

8.1.5 Editor de Archivo de Test

Se trata del editor configurado por defecto para editar cualquier fichero con extensión **.xml**. Consta de tres páginas que permiten editar los archivos de test XML de forma diferente, ya sea mediante el editor de texto (denominada **Fuente**), en estructura de árbol (denominada **Árbol**) o gráficamente (denominada **Diseño**).

Las modificaciones realizadas en cualquiera de las páginas del editor se realizarán automáticamente en las otras dos páginas, ya que el contenido de estas estará siempre sincronizado y actualizado.

8.1.5.1 Fuente

Se trata de la página colocada en tercer lugar, la más común y sencilla de utilizar, ya que se trata de un editor de texto en formato XML. En ella se muestra el contenido real del fichero que se está editando, con un diseño estructurado y resaltado en colores que permite visualizar y modificarlo fácilmente.

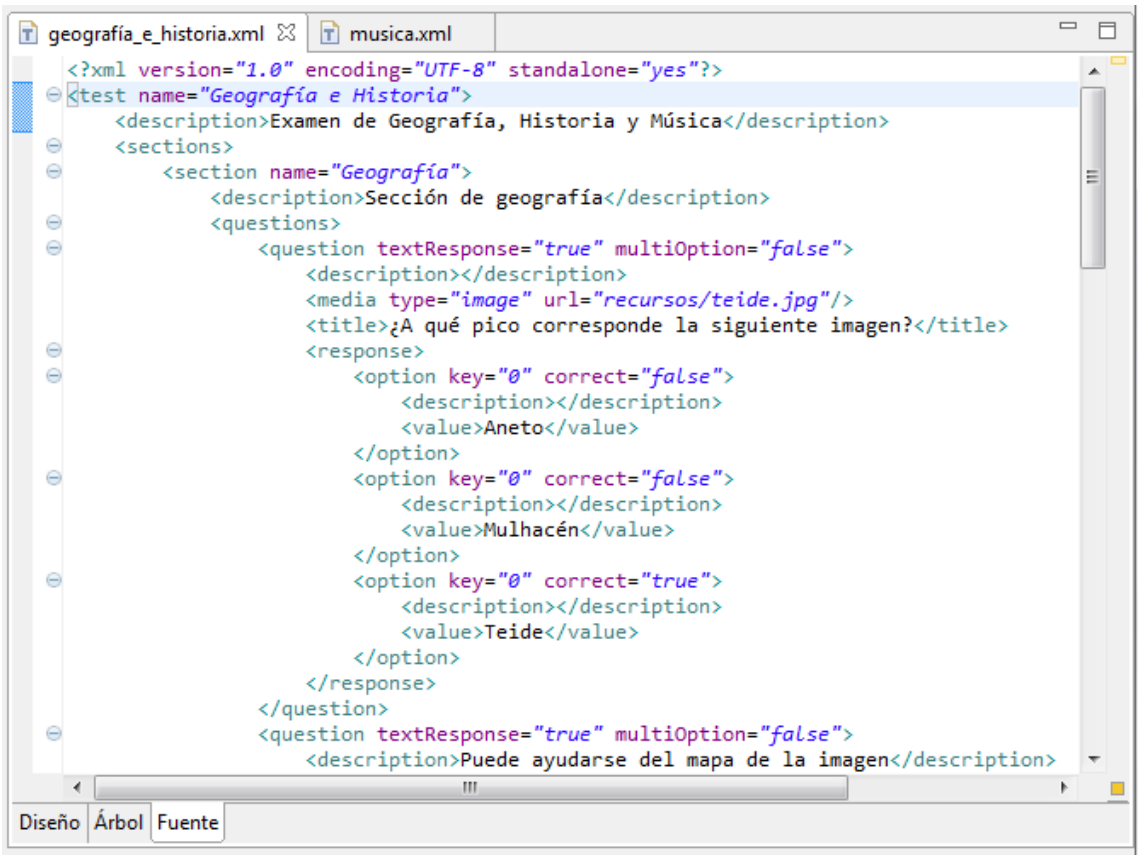


Figura 67 - Página Fuente del editor de test multi-página

Si se desea editar el contenido del fichero mediante esta vista, es posible usar la ayuda contextual (teclas de 'Control '+'Espacio') que muestra los posibles elementos y atributos que se pueden escribir en esa parte del fichero.

Si el contenido mostrado no es correcto y tiene algún error, aparecerá una señal (o muesca) roja (o amarilla en caso de error menor) en los laterales del editor. Si se sitúa el cursor del ratón sobre esta, se mostrará un mensaje con la causa de dicho problema.

Las opciones del menú ***Editar*** descrito en la sección 8.1.3.2 pueden ser utilizadas para editar el contenido, además de las opciones de la ***toolbar*** relativas a la búsqueda dentro del fichero.

Además, el editor consta de un menú contextual que permite realizar acciones como:

- **Revertir** el último cambio o todos los cambios hasta la última versión almacenada.
- **Guardar** el contenido del fichero.

- **Abrir con** otro editor seleccionado de una lista.
- **Mostrar en** una vista seleccionada de una lista.
- **Cortar, copiar** y **pegar** el contenido seleccionado.
- **Fuente:** lista de opciones que permite editar el contenido automáticamente, introduciendo comentarios en formato XML, comentando o eliminando el comentario del texto seleccionado, formateando su contenido, etc.

Abrir la vista de **Propiedades** para visualizar y modificar los valores de los atributos del elemento seleccionado.

- **Validar** el contenido.
- **Trabajo en equipo:** opciones para compartir el fichero, mostrar su historial de versiones y cambios, volver a otra versión anterior, etc.
- **Comparar con:** su historial de versiones (local o remota), otro fichero del repositorio configurado, etc.
- **Sustituir por:** una versión local anterior, otra versión del repositorio configurado, etc.

8.1.5.2 Árbol

La segunda de las páginas del editor muestra en forma de árbol la estructura de nodos que componen el fichero XML.

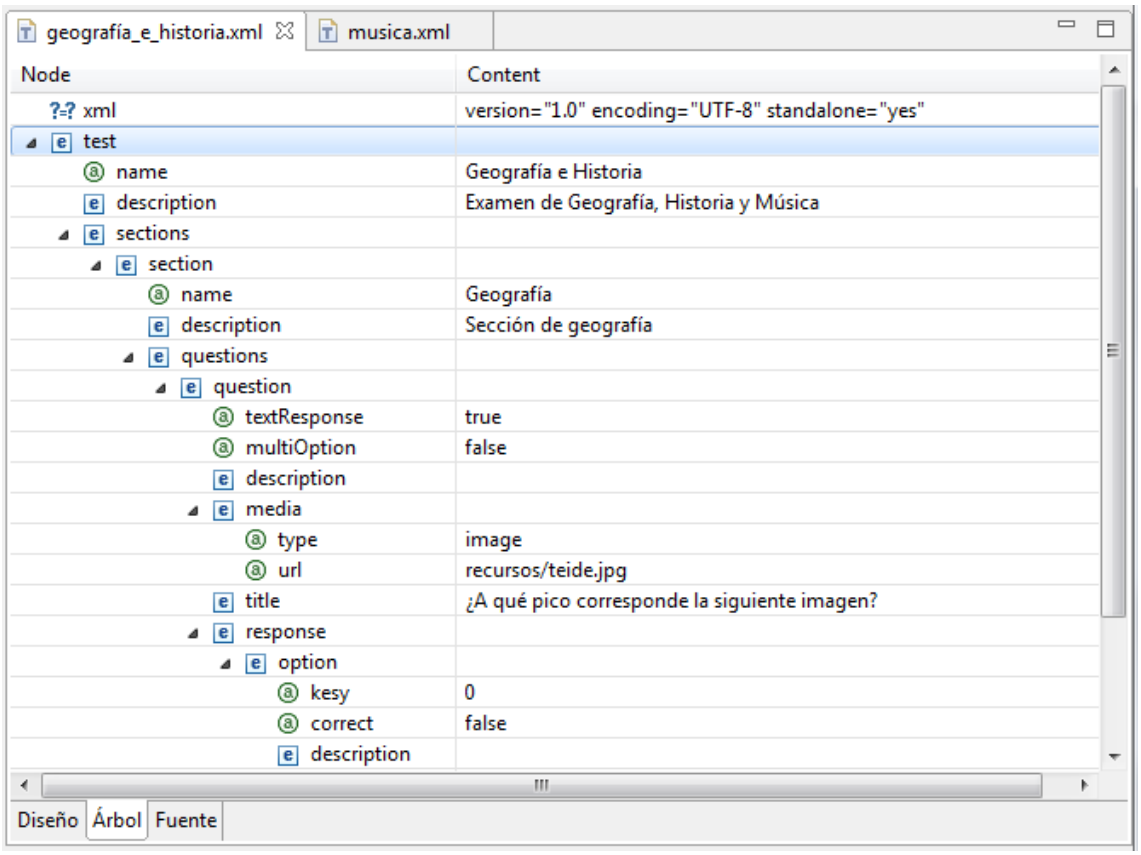


Figura 68 - Página Árbol del editor de test multi-página

Permite hacer modificaciones sobre éste, ya sea:

- cambiando directamente el valor de los atributos en la tabla.
- mediante el menú contextual. Las opciones son:
  - **eliminar** el elemento seleccionado,
  - **reemplazar** el tipo de elemento seleccionado,
  - **añadir** atributos, nuevos hijos, antes o después del elemento seleccionado.
- arrastrando y soltando elementos de un lado a otro. Sin embargo no es aconsejable hacer uso de esta opción a no ser que se tenga conocimiento de qué se está moviendo y a donde, ya que el fichero podría sufrir modificaciones que no cumplan con la estructura lógica del test. En ese caso, la página de *Diseño* mostraría un error y no podría ser utilizada hasta que el formato del fichero volviese a ser correcto.

8.1.5.3 Diseño

Se trata del editor gráfico de pre-visualización del test. Siempre y cuando el contenido textual del fichero sea correcto, aparecerá la estructura de secciones, preguntas y respuestas, con las respectivas imágenes si éstas estuviesen definidas en el fichero; en caso contrario, si el fichero no es correcto, la cabecera del editor mostrará un mensaje de error.

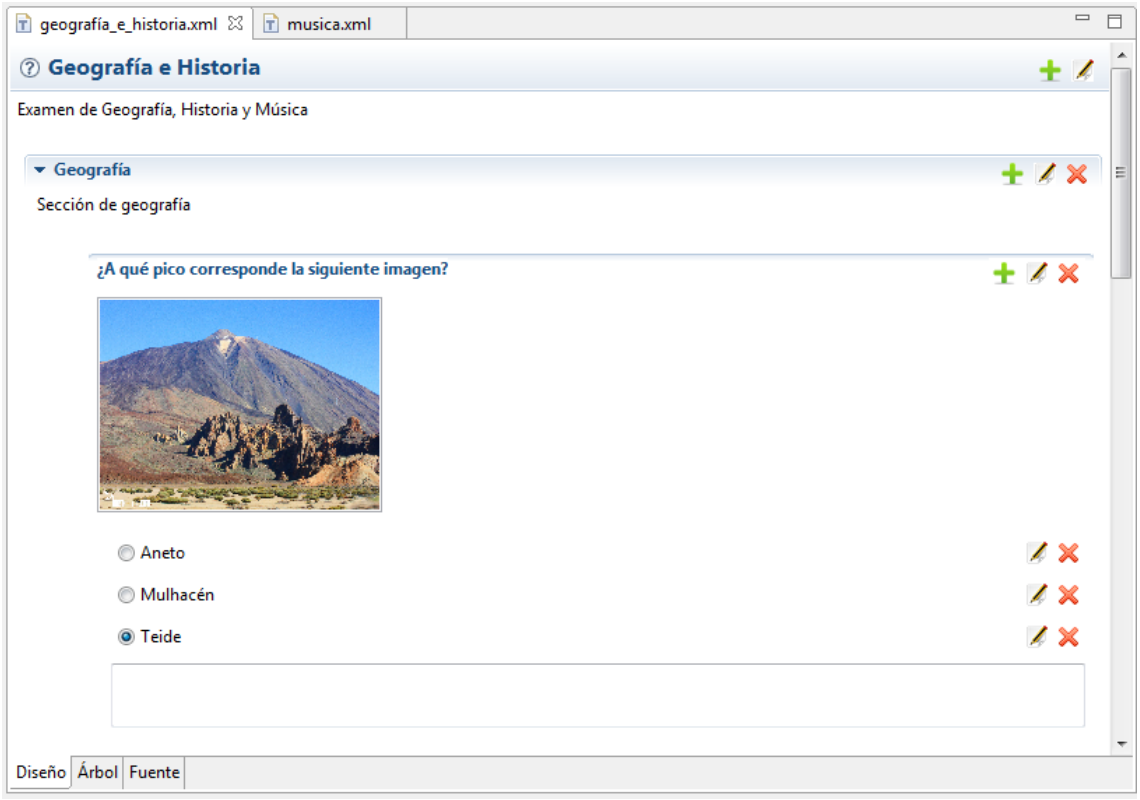


Figura 69 - Página diseño del editor de test multi-página

El editor se divide en dos zonas diferenciadas:

- La primera de ellas y ubicada a la izquierda del editor gráfico, se trata de la parte de representación del fichero de test. En ella se muestran todos los elementos gráficos de los cuales se compone el fichero de test. Permite desplegar y replegar su contenido por secciones, haciendo que la visualización de test largos y complejos se haga más llevadera.

- La segunda y ubicada en el lado derecho, se trata de las opciones que van a permitir, de manera gráfica, editar los componentes definidos en el test. Estas permiten, según el componente en cuestión, añadir nuevos sub-componentes, editarlo o borrarlo de la estructura de test.

A continuación se explican estas opciones y se muestran ejemplos de los diálogos de creación y edición definidos. Dependiendo del nivel en el cual se seleccione una opción, los diálogos serán unos u otros. Estas opciones son:

 **Añadir:** dependiendo de sobre qué componente se elija esta opción, se mostrará un diálogo un diálogo u otro. Es decir, si se elige añadir un componente desde la opción colocada en el título del test, el diálogo abierto permitirá añadir una nueva sección; en cambio si se va a añadir un componente a una sección, el diálogo será el que permite crear una nueva pregunta; mientras que si es a nivel de una pregunta dónde se selecciona esta opción, el diálogo permitirá crear una nueva opción de respuesta a esa pregunta.

 **Editar:** en este caso, la opción elegida si abrirá el diálogo de edición correspondiente al componente sobre el cual se ha pulsado. Así, en la barra de opciones del título del test, esta opción mostrará el diálogo de edición del test, si se hace sobre una sección aparecerá su propio diálogo de edición, etc.

 **Eliminar:** al igual que la anterior, esta opción eliminará el componente sobre el cual se ha seleccionado, previa confirmación mediante un diálogo.

Los diálogos que permiten crear o editar los diferentes tipos de elementos (sección, pregunta, respuesta…) son prácticamente idénticos para todos los tipos. A continuación se muestra uno de ellos, que es el mismo para los elementos de tipo ***test*** y ***sección***:

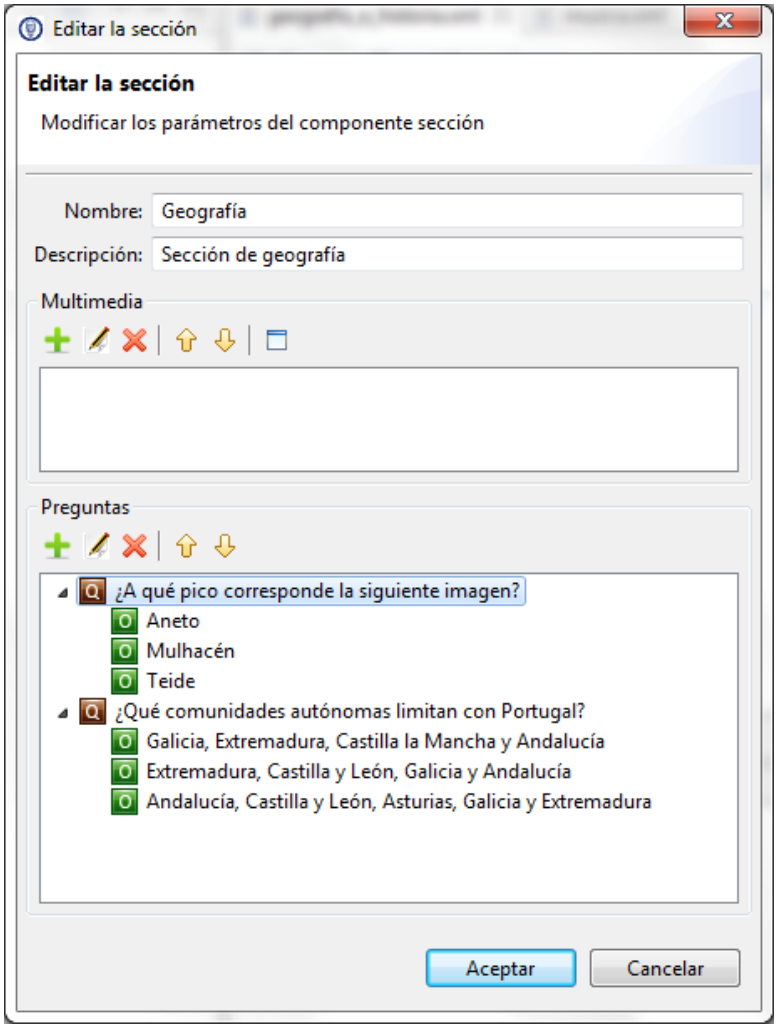


Figura 70 – Diálogo de edición de sección (y de test)

En este tipo de diálogo, se debe configurar obligatoriamente el nombre (o título) del elemento a añadir o editar, además de una descripción.

Dentro del grupo denominado *Multimedia* se muestra la lista de recursos gráficos que posee este componente, y se podrán añadir, modificar o eliminar otros de la lista a través del diálogo (Figura 73) de creación de un nuevo recurso gráfico. También podría ser pre-visualizado desde aquí, mediante la opción definida por el icono , usando el programa por defecto definido en el sistema operativo para su tipo de extensión.

Además, en el grupo inferior del diálogo se muestra un árbol que contiene los elemento hijos del elemento editado, en el que se podrán añadir, modificar o eliminar estos elementos hijos del mismo, como serían secciones a un test, preguntas a una sección y opciones a una pregunta.

Las opciones que muestran tanto la lista de recursos gráficos como el árbol con la estructura de elementos hijos del componente editado son muy similares, con las posibilidades anteriormente descritas de añadir, modificar y eliminar elementos, a las que se añaden dos que dan la posibilidad de intercambiar el orden de sus elementos, y definidas por los botones ⇧ y ⇩.

Además se ha de indicar, que en el caso de edición de un elemento de tipo pregunta u opción, se añaden otros campos y opciones:

- Si se trata del diálogo de edición de tipo ***pregunta***, el diálogo añade dos opciones que permiten definir si la pregunta va a requerir unas opciones de tipo multi-respuesta, o de selección simple con una única opción válida. Además, permite definir si se añade un campo de texto o no a la respuesta.

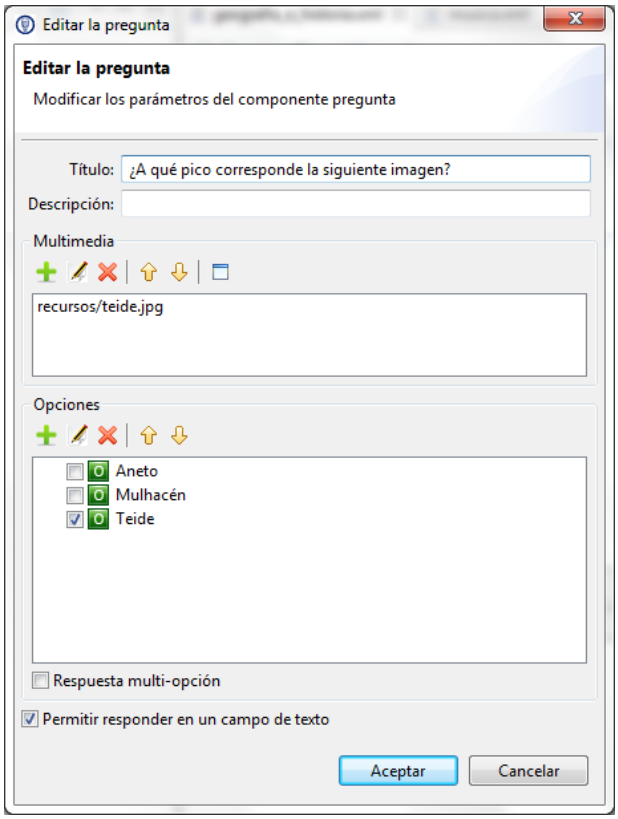


Figura 71 – Diálogo de edición de una pregunta del test

- Si se trata de un diálogo de edición de tipo ***opción***, se añadirá un botón en forma de bandera que permite definir si esa opción es o no válida.

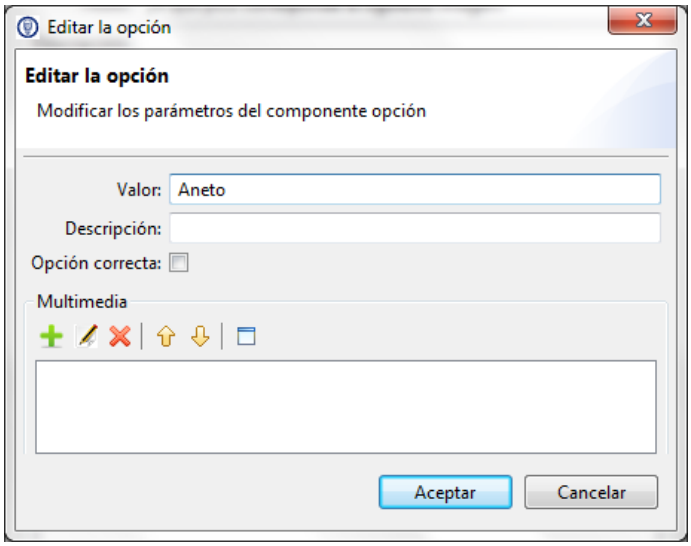


Figura 72 – Diálogo de edición de una opción del test

En cuanto al diálogo de edición de un recurso gráfico, a continuación se muestra su apariencia.

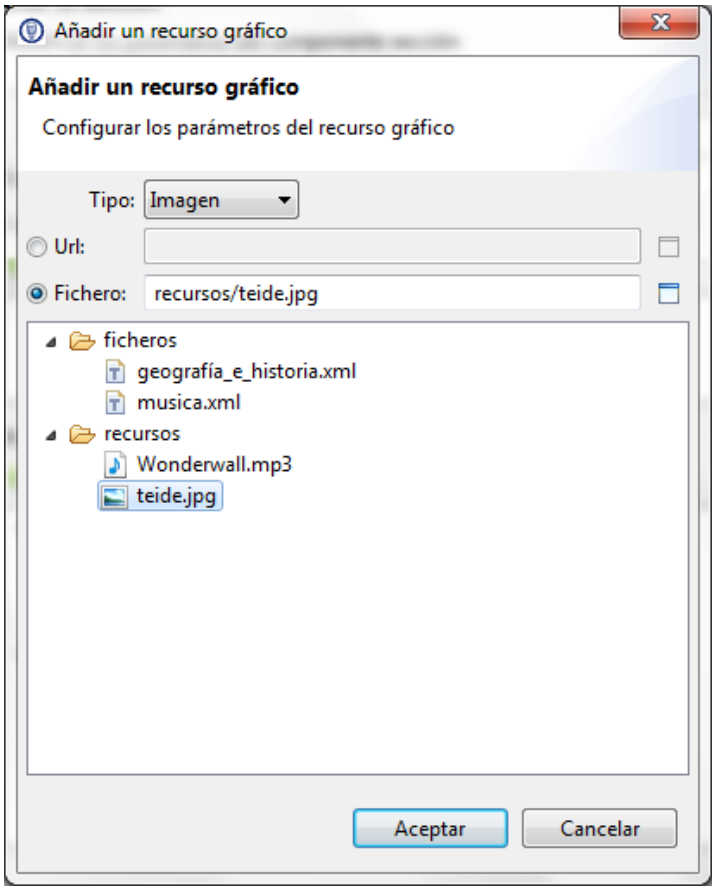


Figura 73 – Diálogo de creación de un nuevo recurso gráfico

En él es posible elegir el tipo de recurso que va a representa, de entre *imagen, audio* o *video*. Y además se debe introducir un valor para este elemento, que bien puede ser una URL a un elemento de la web o una ruta relativa a un elemento disponible en el proyecto al cual pertenece el test editado. Desde aquí pueden ser pre-visualizados los elementos seleccionados mediante la opción definida por el icono  .

8.2 Pruebas

8.2.1 Plan de pruebas manuales

Las pruebas manuales se han dividido por secciones y/o componentes. Se han probado todas y cada una de las funcionalidades implementadas, aunque no todas estén marcadas o documentadas en las tablas que a continuación se muestran resumen todas esas pruebas realizadas.

Ciclo de Vida

Prueba	Descripción	Resultado
Pr_A_1	Ejecución de la aplicación	OK
Pr_A_2	Visualización de pantalla de inicio	OK
Pr_A_3	Creación de espacio de trabajo	OK
Pr_A_4	Selección de espacio de trabajo	OK
Pr_A_5	Visualización de la aplicación	OK
Pr_A_6	Cambiar de espacio de trabajo (opción no recordar): *Archivo > Conmutar espacio de trabajo*	OK
Pr_A_7	Cambiar de espacio de trabajo (opción recordar): *Archivo > Conmutar espacio de trabajo*	OK
Pr_A_8	Cambiar de espacio de trabajo (opción recordar)	OK
Pr_A_9	Reiniciar aplicación: *Archivo > Reiniciar*	OK

Pr_A_10	Cerrar aplicación: *Archivo* > *Salir*	OK
Pr_A_11	Cerrar aplicación: aspa roja de la venta principal	OK

Tabla 14 – Pruebas manuales del ciclo de vida de la aplicación

Navegador

Prueba	Descripción	Resultado
Pr_B_1	Visualización del menú contextual: botón derecho del ratón	OK
Pr_B_2	Diálogo nuevo proyecto de test: *Nuevo* > *Proyecto de test*	OK
Pr_B_3	Selección de plantillas en el diálogo y demás opciones	OK
Pr_B_4	Creación de un nuevo proyecto de test: aparece en el navegador de proyectos, con la estructura elegida	OK
Pr_B_5	Diálogo nuevo archivo de test: *Nuevo* > *Archivo de test*	OK
Pr_B_6	Selección de plantillas en el diálogo y demás opciones	OK
Pr_B_7	Creación de un nuevo archivo de test: aparece en el navegador de proyectos, con la plantilla elegida	OK
Pr_B_8	Creación de un nuevo directorio: *Nuevo* > *Carpeta*	OK
Pr_B_9	Creación de un nuevo proyecto: *Nuevo* > *Proyecto*	OK
Pr_B_10	Creación de un nuevo fichero: *Nuevo* > *Archivo*	OK
Pr_B_11	Diálogo de selección de cualquier diálogo de creación de recursos: *Nuevo* > *Otros*	OK
Pr_B_12	Copiar, pegar, suprimir, mover y renombrar un proyecto, un directorio y un archivo: *Nuevo* > *(opción)*	OK
Pr_B_13	Importar un proyecto de test: *Nuevo* > *Importar*, seleccionar	OK

	Proyectos existentes en el espacio de trabajo	
Pr_B_14	Importar un archivo de test: *Nuevo* > *Importar*, seleccionar *Sistema de archivos*	OK
Pr_B_15	Exportar un proyecto o archivo de test: *Nuevo* > *Exportar*, seleccionar *Sistema de archivos*	OK
Pr_B_16	Refrescar el navegador de proyectos	OK
Pr_B_17	Compartir un proyecto con un repositorio de contenidos pre-configurado: *Nuevo* > *Trabajo en Equipo* > *Compartir proyecto*	OK
Pr_B_18	Comparar un proyecto con el almacenado en el repositorio: *Nuevo* > *Comparar con* > *Último almacenado*	OK
Pr_B_19	Bajarse un proyecto desde un repositorio de contenidos pre-configurado: *Nuevo* > *Trabajo en Equipo* > *Descargar*	OK
Pr_B_20	Generar un número elevado de proyectos en el navegador: crear, importar o descargar	OK
Pr_B_21	Crear conjuntos de trabajo	OK
Pr_B_22	Repartir proyectos en conjuntos de trabajo	OK
Pr_B_23	Modificar los conjuntos de trabajo	OK
Pr_B_24	Abrir un archivo de test: doble click del ratón	OK
Pr_B_25	Abrir un archivo de test: *Nuevo* > *Abrir*	OK
Pr_B_26	Abrir un archivo de test con otro editor: *Nuevo* > *Abrir con* > *(opción)*	OK
Pr_B_27	Abrir cualquier otro recurso: imagen, audio, video o fichero no XML	OK

Pr_B_28	Cerrar un proyecto	OK

Tabla 15 – Pruebas manuales del navegador de proyectos de la aplicación

Editor XML Multi-página

Prueba	Descripción	Resultado
Pr_C_1	Apertura de archivo de test XML en editor	OK
Pr_C_2	Visualización de página *Diseño* en primera instancia	OK
Pr_C_3	Visualización correcta si el archivo es correcto	OK
Pr_C_4	Visualización con error si el archivo no es correcto	OK
Pr_C_5	Diálogo de edición del test: *toolbar > Editar Test*	OK
Pr_C_6	Edición de test correcta: nombre, descripción, secciones…	OK
Pr_C_7	Añadir recurso gráfico al test: imagen, audio o video definido en el proyecto	OK
Pr_C_8	Añadir recurso gráfico al test: imagen, audio o video definido mediante URL	OK
Pr_C_9	Visualización de recurso gráfico: imagen definida en el proyecto o recurso mediante doble click	OK
Pr_C_10	Diálogo añadir una sección: *toolbar > Añadir sección*	OK
Pr_C_11	Añadir una sección: visualización correcta	OK
Pr_C_12	Diálogo de edición del sección: *toolbar > Editar Sección*	OK
Pr_C_13	Edición de sección correcta: nombre, descripción, preguntas…	OK
Pr_C_14	Eliminar una sección: *toolbar > Eliminar sección*	OK

Pr_C_15	Diálogo añadir una pregunta: *toolbar > Añadir pregunta*	OK
Pr_C_16	Añadir una pregunta: visualización correcta	OK
Pr_C_17	Diálogo de edición del pregunta: *toolbar > Editar Pregunta*	OK
Pr_C_18	Edición de pregunta correcta: título, descripción, recursos gráficos, respuestas…	OK
Pr_C_19	Eliminar una pregunta: *toolbar > Eliminar pregunta*	OK
Pr_C_20	Diálogo añadir una opción: *toolbar > Añadir opción*	OK
Pr_C_21	Añadir una opción: visualización correcta	OK
Pr_C_22	Diálogo de edición del opción: *toolbar > Editar Opción*	OK
Pr_C_23	Edición de opción correcta: valor, descripción, recursos gráficos o si es o no válida	OK
Pr_C_24	Eliminar una opción: *toolbar > Eliminar opción*	OK
Pr_C_25	Repliegue y despliegue de secciones	OK
Pr_C_26	Representación en el editor de árbol	OK
Pr_C_27	Edición en el editor de árbol	OK
Pr_C_28	Despliegue y repliegue en el editor de árbol	OK
Pr_C_29	Representación en el editor de fuente XML	OK
Pr_C_30	Edición en el editor de fuente XML	OK
Pr_C_31	Visualización de errores en el editor de fuente XML	OK
Pr_C_32	Opción de autocompletar en el editor de fuente XML	OK

Pr_C_33	Opciones del menú contextual del editor de fuente XML: cortar, copiar, pegar, etc.	OK
Pr_C_34	Visualización de la selección en la vista de *Propiedades*	OK
Pr_C_35	Visualización de la estructura del fichero en la vista *Esquema*	OK
Pr_C_36	Visualización de la selección en la barra inferior de la aplicación	OK

Tabla 16 – Pruebas manuales del editor multi-página de la aplicación

Opciones de menú y toolbar

Prueba	Descripción	Resultado
Pr_D_1	Opción de menú y toolbar: *Archivo > Nuevo > (opción)*	OK
Pr_D_2	Opción de menú y toolbar: *Archivo > Guardar*	OK
Pr_D_3	Opción de menú y toolbar: *Archivo > Guardar como*	OK
Pr_D_4	Opción de menú y toolbar: *Archivo > Imprimir*	OK
Pr_D_5	Opciones de la toolbar de edición y búsqueda de texto	OK
Pr_D_6	Cerrar el editor actual: *Archivo > Cerrar*	OK
Pr_D_7	Cerrar todos los editores abiertos: *Archivo > Cerrar todos*	OK
Pr_D_8	Opción de mover el recurso seleccionado en el navegador: *Archivo > Mover*	OK
Pr_D_9	Opción de renombrar el recurso seleccionado en el navegador: *Archivo > Renombrar*	OK
Pr_D_10	Opción de refrescar el navegador: *Archivo > Renovar*	OK
Pr_D_11	Importar un proyecto de test: *Archivo > Importar,*	OK

	seleccionar *Proyectos existentes en el espacio de trabajo*	
Pr_D_12	Importar un archivo de test: *Archivo* > *Importar*, seleccionar *Sistema de archivos*	OK
Pr_D_13	Exportar un proyecto o archivo de test: *Archivo* > *Exportar*, seleccionar *Sistema de archivos*	OK
Pr_D_14	Opciones del menú *Editar* contra el editor de texto: deshacer, rehacer, copiar, cortar, pegar, borrar, seleccionar todo y buscar/remplazar	OK
Pr_D_15	Opciones del menú *Editar* contra el navegador de proyectos: deshacer, rehacer, copiar, pegar, borrar y seleccionar todo	OK
Pr_D_16	Diálogo de búsqueda: *Buscar* > *Abrir diálogo de buscar*	OK
Pr_D_17	Realizar búsqueda desde diálogo: en todo el espacio de trabajo, solo en ese proyecto o solo en el recurso seleccionado	OK
Pr_D_18	Realizar búsqueda del texto seleccionado en el espacio de trabajo	OK
Pr_D_19	Realizar búsqueda del texto seleccionado en el proyecto	OK
Pr_D_20	Realizar búsqueda del texto seleccionado en el archivo	OK
Pr_D_21	Realizar búsqueda del texto seleccionado en el conjunto de trabajo	OK
Pr_D_22	Ocultar la toolbar: *Ventana* > *Ocultar barra de herramientas*	OK
Pr_D_23	Seleccionar y abrir una vista del diálogo: *Ventana* > *Mostrar Vista*	OK
Pr_D_24	Seleccionar y abrir una perspectiva del diálogo: *Ventana* > *Mostrar Perspectiva*	OK

Pr_D_25	Reiniciar la perspectiva a su estado por defecto: *Ventana* > *Restablecer Perspectiva*	OK
Pr_D_26	Abrir la ventana externa con el manual de usuario: *Ayuda* > *Ayuda*	OK
Pr_D_27	Abrir la vista de ayuda interna en la aplicación: *Ayuda* > *Buscar ayuda* o *Ayuda* > *Ayuda dinámica*	OK
Pr_D_28	Diálogo con los datos de instalación: *Ayuda* > *Acerca de*	OK

Tabla 17 – Pruebas manuales de las opciones de menú y toolbar de la aplicación

8.3 Ficheros de configuración

8.3.1 Fichero XSD usado por el editor XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="media" type="media"/>

  <xs:element name="option" type="option"/>

  <xs:element name="question" type="question"/>

  <xs:element name="section" type="section"/>

  <xs:element name="test" type="test"/>

  <xs:complexType name="test">
    <xs:complexContent>
      <xs:extension base="abstractComponent">
        <xs:sequence>
          <xs:element name="sections" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:element ref="section" minOccurs="0" maxOccurs="unbounded"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
<xs:complexType name="abstractComponent" abstract="true">
  <xs:sequence>
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <xs:element ref="media" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="section">
  <xs:complexContent>
    <xs:extension base="abstractComponent">
      <xs:sequence>
        <xs:element name="questions" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
             <xs:element ref="question" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="question">
  <xs:complexContent>
    <xs:extension base="abstractComponent">
      <xs:sequence>
        <xs:element name="title" type="xs:string" minOccurs="0"/>
        <xs:element name="response" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="option" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="textResponse" type="xs:boolean" use="required"/>
      <xs:attribute name="multiOption" type="xs:boolean" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="option">
  <xs:complexContent>
    <xs:extension base="abstractComponent">
      <xs:sequence>
        <xs:element name="value" type="xs:string" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="key" type="xs:int" use="required"/>
      <xs:attribute name="correct" type="xs:boolean" use="required"/>
    </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="media">
    <xs:sequence/>
    <xs:attribute name="type" type="xs:string"/>
    <xs:attribute name="url" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

8.4 Crear un plugin de Eclipse

A continuación se explica cómo proceder a la hora de crear un nuevo plugin de Eclipse.

Decir que se pueden crear diferentes tipos de plugin, dependiendo de la finalidad que se le quiera dar. Esto se configura en una de las páginas del diálogo de creación de un nuevo plugin, y existen cuatro posibilidades:

- Aplicación independiente con entorno gráfico.
- Aplicación independiente sin entorno gráfico.
- Funcionalidad adicional aportada a un entorno gráfico.
- Funcionalidad adicional simple, sin aporte gráfico.

Para crear un plugin, se deben seguir los siguientes pasos:

- **Paso 1:** abrir el diálogo de creación de un nuevo plugin, desde la opción *Nuevo > Plug-in Project* disponible en el menú superior *File,* en la toolbar o en el menú contextual del navegador de proyectos.

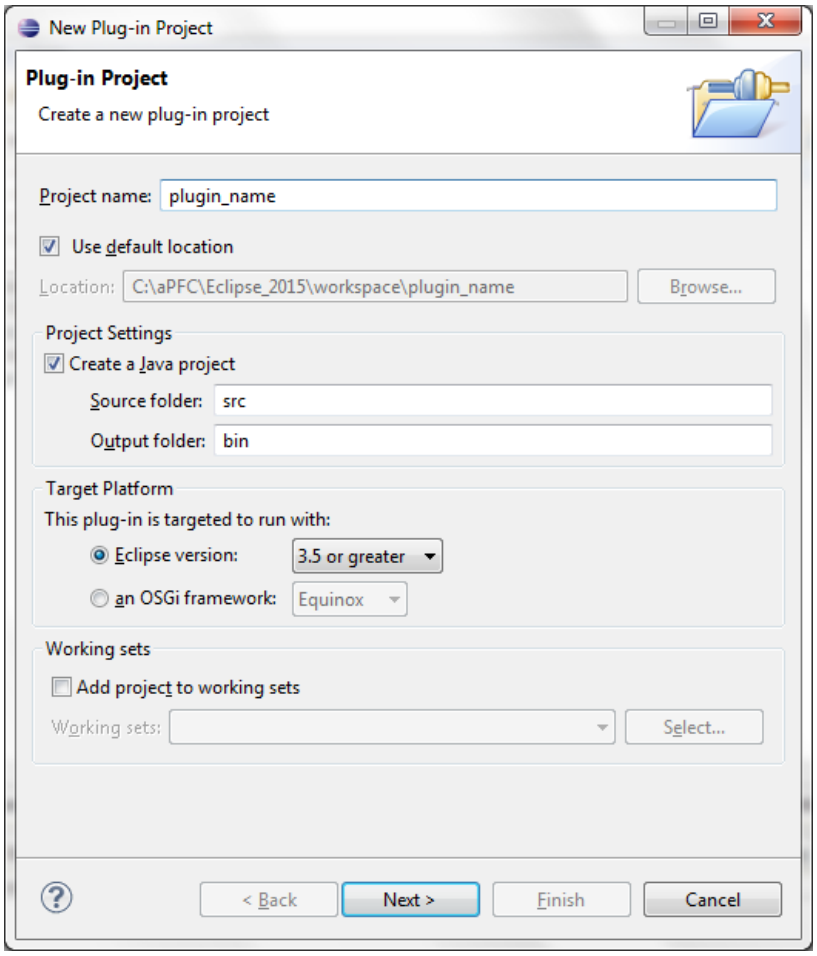


Figura 74 – Diálogo de creación de un nuevo de plugin (primera página)

- **Paso 2:** introducir el nombre del plugin, y los demás datos si fuese preciso.

- **Paso 3:** pulsar *Next* para pasar a la siguiente página del diálogo.

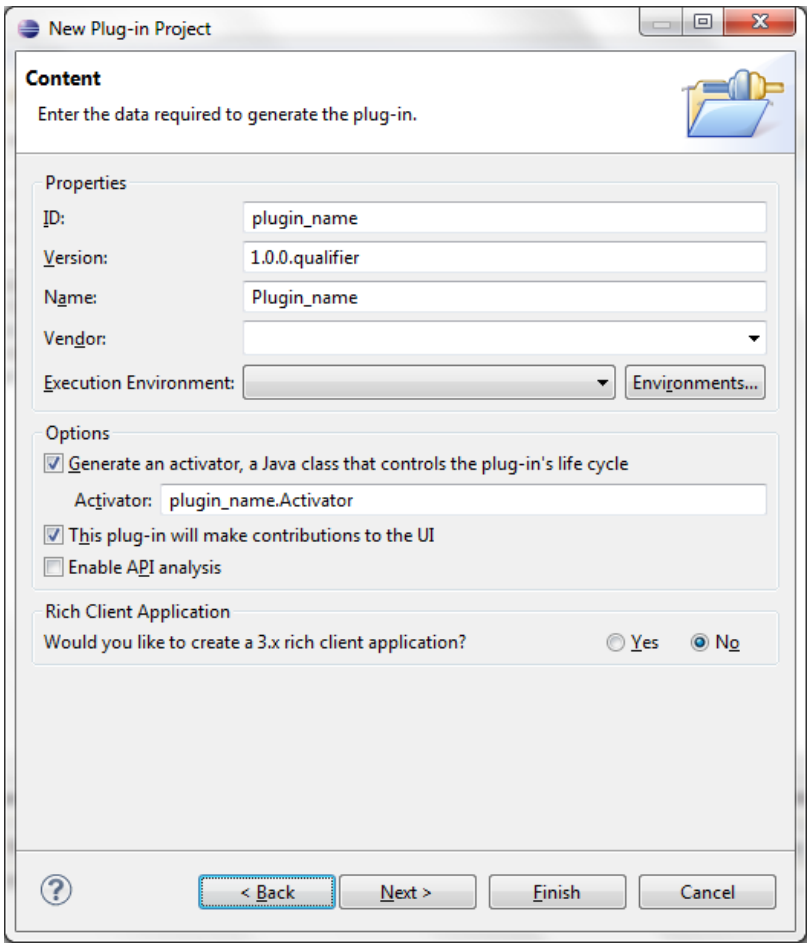


Figura 75 – Diálogo de creación de un nuevo de plugin (segunda página)

- **Paso 4:** en la nueva página del diálogo, introducir los datos en *Properties*.

- **Paso5:** seleccionar o no, aunque se recomienda, la opción generar un activador para el plugin, que consiste en crear una clase que se denominada *Activator.java* que será quien inicialice el plugin al ser instanciada.

- **Paso 6:** según el tipo de plugin a crear de los anteriormente enumerados, se deberá escoger entre las opciones *si este plugin es o no una contribución gráfica* y/o si *se desea crear un aplicación RCP*.

- **Paso 7:** seleccionar un tipo y pulsar *Next* para visualizar la siguiente página del diálogo.

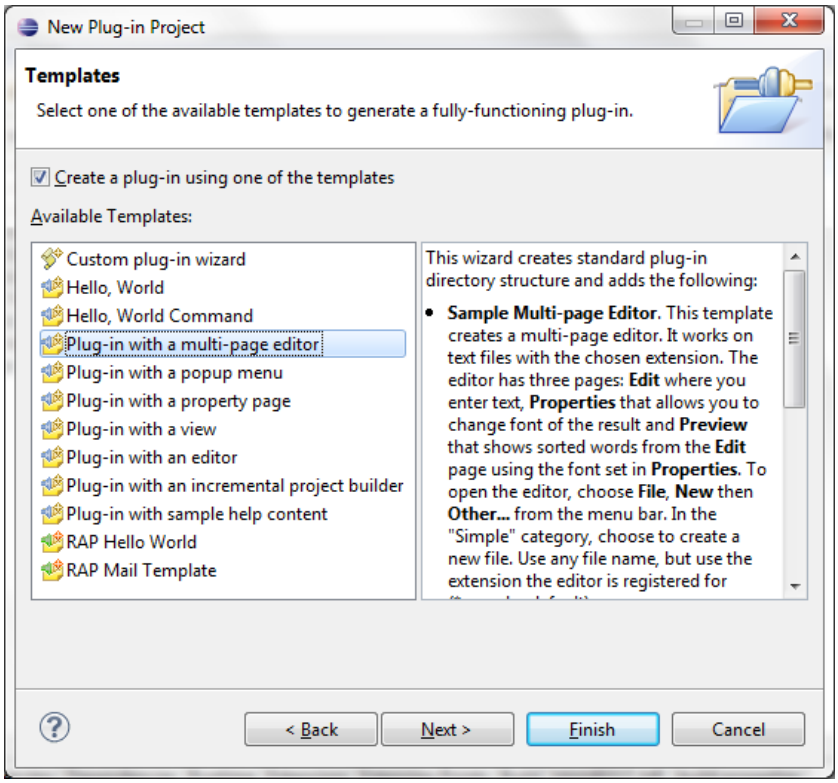


Figura 76 – Diálogo de creación de un nuevo de plugin (tercera página)

- **Paso 8:** en función del tipo de plugin a crear elegido, variarán las plantillas que aparecen en el diálogo.

- **Paso 9:** elegir una y pulsar la opción *Next* para pasar a la cuarta página del diálogo de creación de un nuevo plugin.

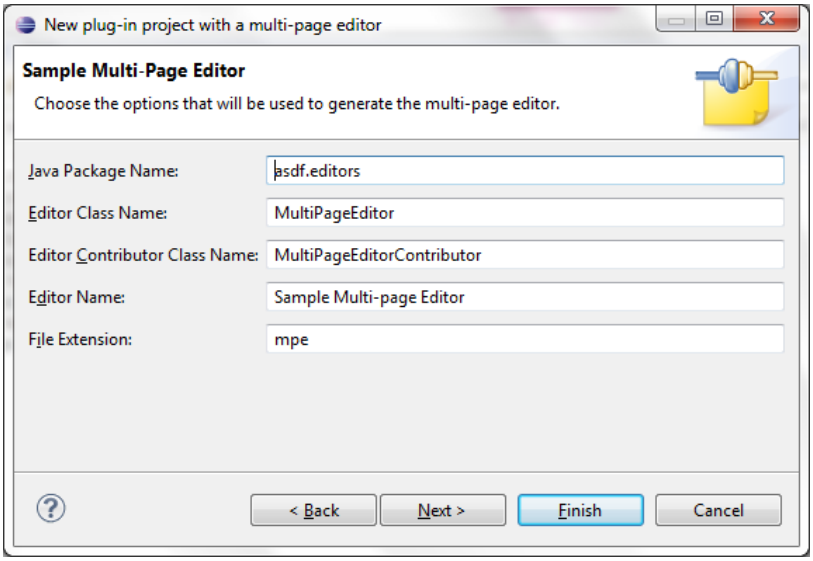


Figura 77 - Diálogo de creación de un nuevo de plugin (cuarta página)

- **Paso 10:** en esta página aparecerán diferentes datos dependiendo del tipo de plantilla elegida. Introducir los datos.

- **Paso 11:** de nuevo dependiendo de la plantilla seleccionada, ésta será la última página del diálogo o podrían existir otras que permitan configurar otro tipo de parámetros. En todo caso, se puede proceder a pulsar *Finish* para así finalizar la creación del nuevo proyecto de plugin.

8.4.1 Internacionalizar un plugin de Eclipse

Los pasos para internacionalizar una aplicación Eclipse RCP serán:

1. Para llevar a cabo esta opción, lo primero es descargar los plugins complementarios de internacionalización que ya existen sobre SWT, JFace, y todos los demás plugins de Eclipse incluidos en la distribución de RCP.

   Se encuentran disponibles en la página web:

   https://eclipse.org/babel/downloads.php

2. Posteriormente habrá que definir los ficheros de propiedades con todos los valores clave/valor que contienen las etiquetas que se van a transformar. Estos ficheros de propiedades se llamaran por ejemplo:

   - messages.properties: fichero de language por defecto.
   - messages_es.properties: para el español.
   - messages_en.properties: por defecto para el Inglés.

   Estos ficheros tendrán la estructura clave/valor:

```
# messages.properties file

View_0=Overview

View_1=Details
```

3. Estas claves son el valor que se va a definir en el fichero de configuración que extenderá la clase 'org.eclipse.osgi.util.NLS':

```
import org.eclipse.osgi.util.NLS;

public class Messages extends NLS {

    private static final String BUNDLE_NAME = "messages";
```

```
        public static String View_0;
        public static String View_1;
        static {
                // initialize resource bundle
                NLS.initializeMessages(BUNDLE_NAME, Messages.class);
        }
        private Messages() {  }
    }
```

4. Estas claves son los mensajes a usar en las etiquetas definidas en los widgets (SWT, JFace o Eclipse Forms).

```
label.setText(Messages.View_1);
```

5. Para configurar una etiqueta en el fichero *plugin.xml*, se ha de configurar con el carácter '%' delante. Es decir:

```
label="%View_1"
```

6. Además, será necesario añadir una línea en el fichero *MANIFEST.MF* con la siguiente etiqueta y la ruta relativa en el proyecto donde encontrar los ficheros de propiedades de los mensajes. Será algo como:

```
Bundle-Localization: messages
```

7. Por último, falta elegir el lenguaje con el cual se desea configurar la aplicación. Esto se puede probar con el argumento '-nl' a la hora de arrancar la aplicación:

```
-nl es
```

8.5 Configuración de Eclipse

8.5.1 Formatter y Clean-up

Eclipse separa en dos partes las herramientas que proporciona por defecto para hacer que el código sea lo mayor estructurado posible y guarde unas reglas de calidad mínimas. Estas partes son la limpieza y el formato.

Se puede ejecutar cada uno por separado, pero se aconseja realizar ambas opciones conjuntamente, ya que tiene mejores resultados y al final de lo que se trata es de tener un código lo más legible posible, ya que nunca se sabe quién va a tener que leer, comprender y realizar modificaciones sobre este.

Además, estas opciones se pueden automatizar, de manera que se ejecuten cada vez que se salva el contenido del fichero que se está editando. Esto se hace es la ruta:

Window > Preferences > Java > Editor > Save Actions

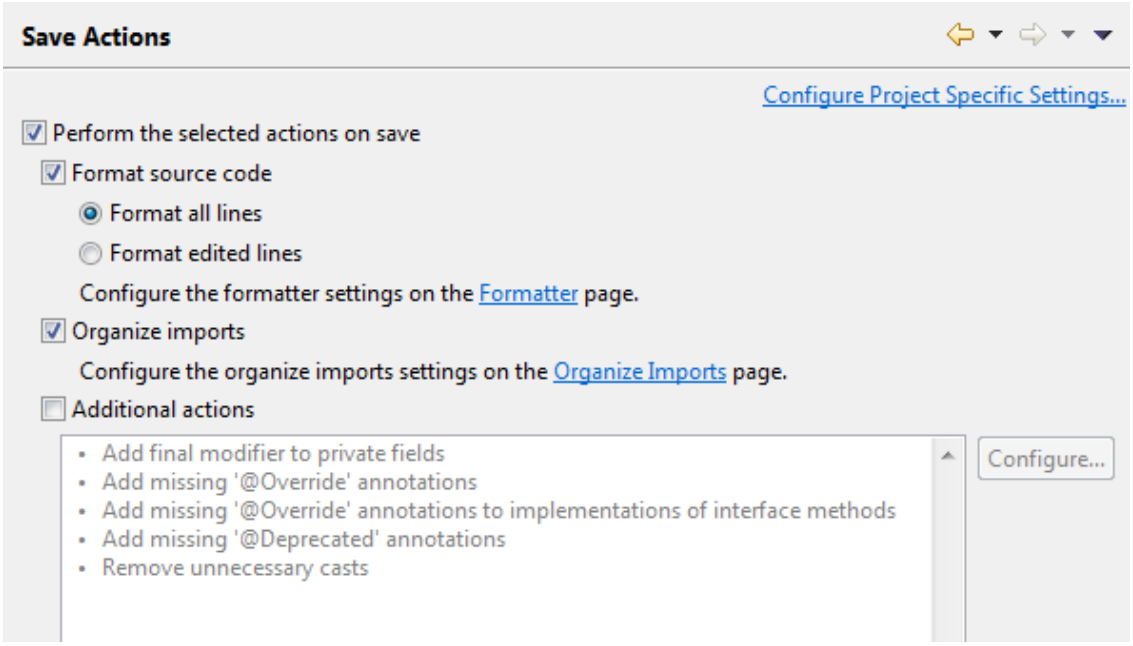


Figura 78: panel de 'Save Actions'

Dentro de Eclipse, la opción de '**Clean-up**' se configura en la ruta:

Window > Preferences > Java > Code Style > Clean Up

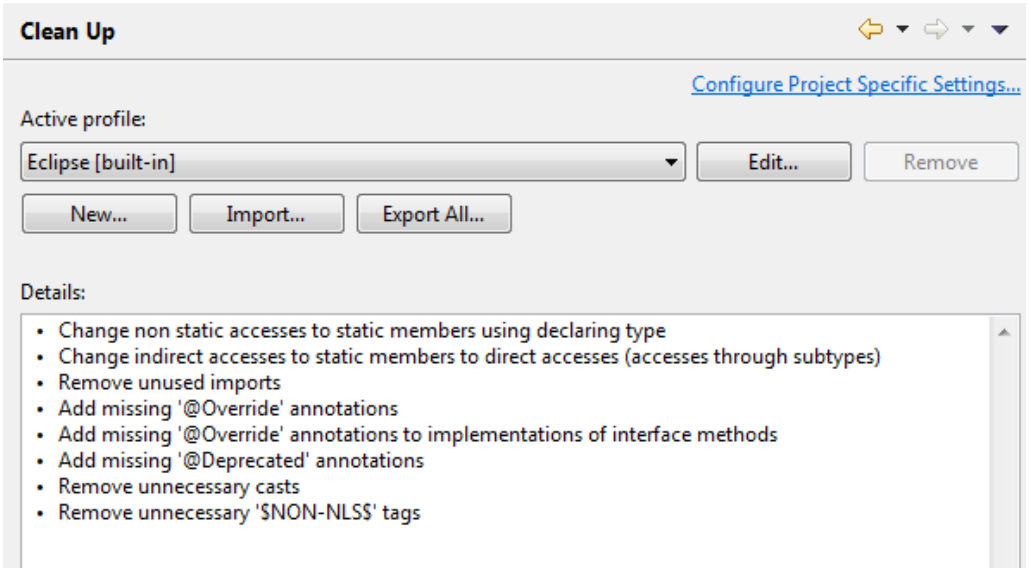


Figura 79: Panel de 'Clean Up'

Y en esa misma ruta, encontramos la opción '**Formatter'**:

Window > Preferences > Java > Code Style > Formatter

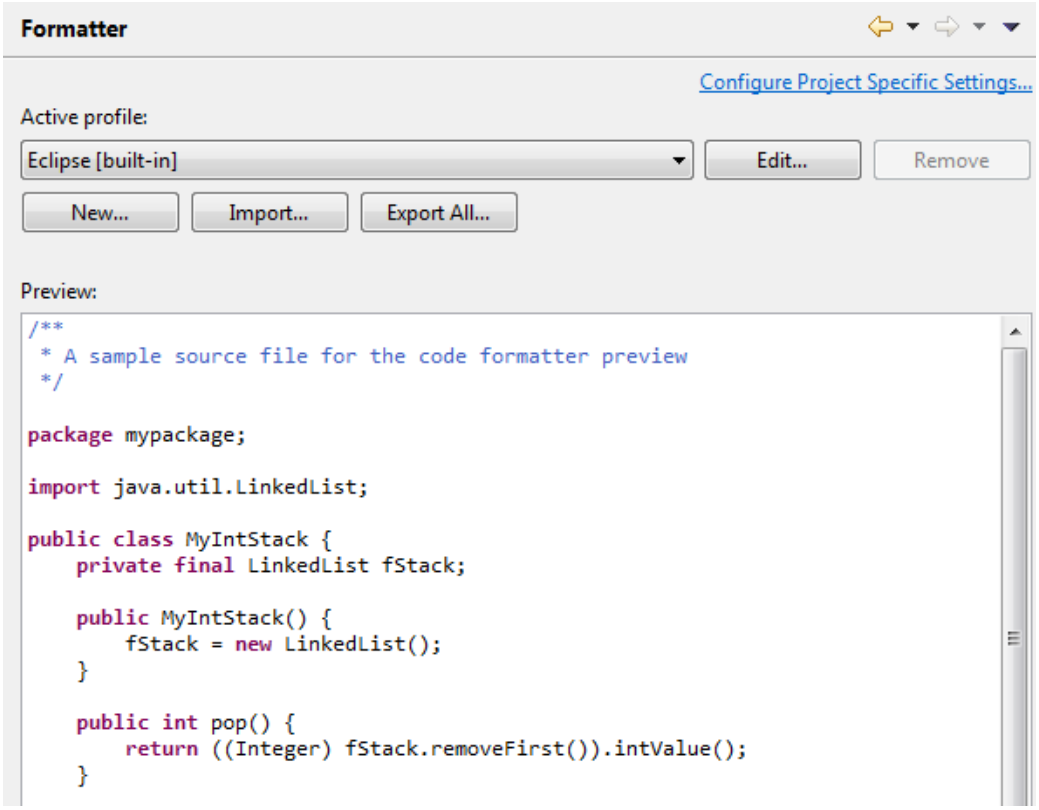


Figura 80: Panel de 'Formatter'

8.5.2 Plugin Checkstyle

Este plugin no forma parte por defecto de ninguna de las distribuciones de Eclipse. Se ha de añadir desde las opciones de Eclipse o bien descargándolo desde la página web y añadiéndolo a la carpeta '/plugins' de la instalación.

Dentro de Eclipse, las opciones de Checkstyle se configuran en la ruta:

Window > Preferences > Checkstyle

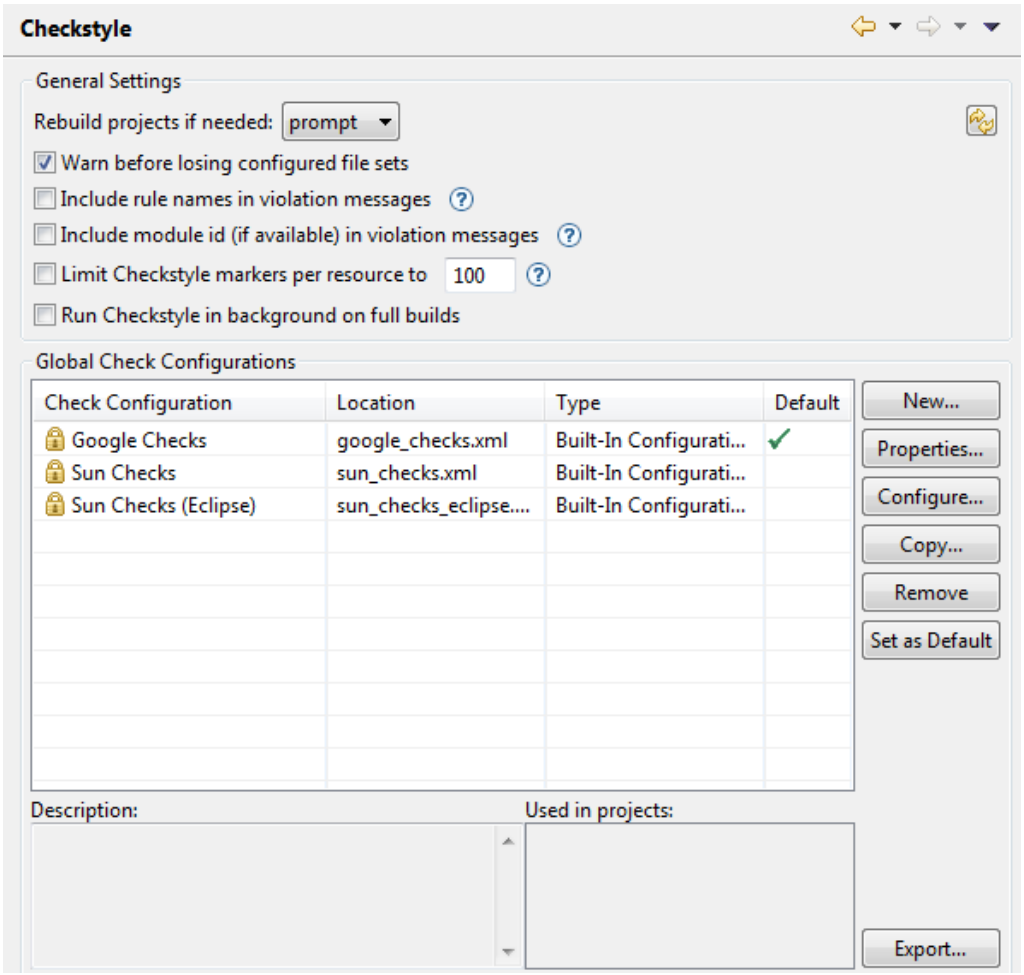


Figura 81: Panel de 'Checkstyle'